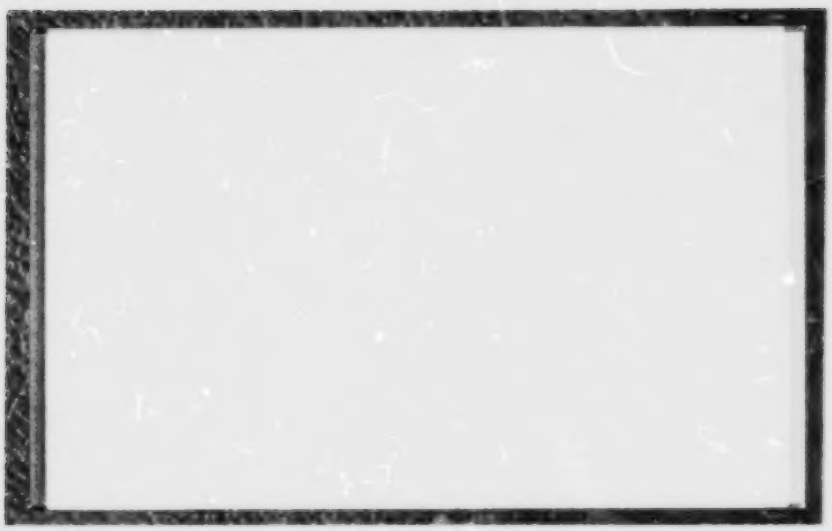
UNIVERSITÉ DE MONTRÉAL



CENTRE DE RECHERCHE SUR LES TRANSPORTS

MÉMOIRE D'ÉTUDIANT

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

RÉSOLUTION NUMÉRIQUE DU PROBLÈME DE MULTIFLOT COMPATIBLE À COÛT MINIMUM

par

Michel Sauvé

Mémoire de M.Sc., département d'informatique et de recherche opérationnelle, dirigé par Sang Nguyen

Université de Montréal
Centre de recherche sur les transports - Publication #409
Octobre 1984

Dépôt légal - 2e trimestre 1985

TABLE DES MATIERES

LISTE DES FIGURES vi

LISTE DES TABLEAUX viii

SOMMAIRE ix

CHAPITRE PREMIER: INTRODUCTION 1

1.1 La classe des modèles de multiflot 1

1.2 Le modèle de multiflot compatible à valeur maximale 3

1.3 Le modèle de multiflot d'équilibre 4

1.4 Le modèle de multiflot compatible à coût minimum 5

1.5 Problématique 6

CHAPITRE DEUXIEME: L'ETAT DES RECHERCHES 8

2.1 Une revue des méthodes de solution 8

2.1.1 Décomposition de Dantzig-Wolfe 9

2.1.2 Partition primale de Rosen 15

2.1.3 Allocation de ressource 22

2.1.4 Résolution du dual par la méthode de Bender 34

2.1.5 Generalized Upper Bounding 39

2.1.6 La méthode du lagrangien augmenté 52

2.1.7 Les méthodes de direction de descente pour le modèle de multiflot d'équilibre 58

2.2 Analyse comparative 80

2.2.1 Propriété théorique 80

2.2.2 Résultats numériques 85

2.2.3 Conclusion 91

CHAPITRE TROISIEME: UN ALGORITHME POUR LA RESOLUTION DU PROBLEME DE MULTIFLOT COMPATIBLE A COUT MINIMUM

3.1 L'ajustement des multiplicateurs 95

3.1.1 Pondération des contraintes de capacité 95

3.1.2 L'algorithme 96

3.1.3 Solution initiale 98

3.1.4 Critère d'arrêt 98

3.1.5 Sélection de $\overline{k}$ 98

3.2 Le simplexe convexe 103

3.2.1 La description du réseau 104

3.2.2 Structure de base et opérations sur la base 106

3.2.3 L'algorithme 131

CHAPITRE QUATRIEME: RESULTATS NUMERIQUES 141

4.1 Nature des problèmes 142

4.2 Environnement 145

4.3	Calcul de multiflot d'équilibre	145
	4.3.1 Le problème de Hull	146
	4.3.2 Le problème de Winnipeg	150
4.4	Choix d'un vecteur de capacité	152
4.5	Résolution des problèmes MCM	154
	4.5.1 Les problèmes HULβ	154
	4.5.2 Les problèmes WIN10	156
4.6	Conclusion	171
	4.6.1 Le simplexe convexe appliqué au multiple d'équilibre	171
	4.6.2 Le lagrangien augmenté et le modèle MCM	175
CONCLUSION		177
BIBLIOGRAPHIE		180
REMERCIEMENTS		

LISTE DES FIGURES

3.1 Choix du prochain flot

3.2 Représentation du réseau: exemple

3.3 Représentation de la base: exemple

3.4 Modification du flot sur l'arc A: organigramme

3.5 Traçage de cycle: exemple graphique

3.6 Mise-à-jour des potentiels: organigramme

3.7 Micro-pivot: organigramme

3.8 Notation graphique de M.-A. Denis: exemple

3.9 Méthode de pivotage: exemple graphique

3.10 Méthode de pivotage: exemple graphique

3.11 Méthode de pivotage: exemple graphique

3.12 Méthode de pivotage: exemple graphique

3.13 Ajout d'un chemin à la fonction de préordre: organigramme

3.14 Concergence lente pour "regula falsi": exemple

3.15 "Regula falsi" + bissection: exemple

4.1 HUL-temps de résolution

4.2 WIN-temps de résolution

4.3 HUL2-lagrangien augmenté vs temps

4.4 HUL2-distance pondérée vs temps

4.5 HUL4-lagrangien augmenté vs temps

4.6 HUL4-distance pondérée vs temps

4.7 HUL6-lagrangien augmenté vs temps

4.8 HUL6-distance pondérée vs temps

4.9 HUL8-lagrangien augmenté vs temps

4.10 HUL8-distance pondérée vs temps

4.11 HUL10-lagrangien augmenté vs temps

4.12 HUL10-distance pondérée vs temps

4.13 HUL12-lagrangien augmenté vs temps

4.14 HUL12-distance pondérée vs temps

4.15 WIN10-lagrangien augmenté vs temps

4.16 WIN10-distance pondérée vs temps

LISTE DES TABLEAUX

4.1 Caractéristiques des problèmes

4.2 Ventilation du temps d'exécution

4.3 Résultats des problèmes HULβ

SOMMAIRE

Le modèle de multiflot compatible à coût minimum (Multicommodity Network Flow) et ses variantes a de nombreuses applications dans le domaine des transports et des communications. Cependant, les problèmes réels nécessitent l'optimisation de modèles de grande taille. Le but de ce travail est la construction d'un algorithme efficace pour résoudre de tels modèles.

Nous présentons d'abord une revue des méthodes de résolution proposées dans la littérature. Nous discutons leurs propriétés et l'expérience numérique accessible.

Nous présentons ensuite une heuristique basée sur la relaxation des contraintes de capacité. Cet algorithme utilise le schéma proposé par Van der Straatten-Tilquin pour l'application de la méthode du lagrangien augmenté. On obtient ainsi une suite de problèmes de multiflot d'équilibre qui sont résolus à l'aide d'une spécialisation du simplexe convexe proposée par Nguyen. Une implantation efficace a été réalisée.

Nous avons d'abord appliqué l'algorithme de Nguyen à des problèmes réels de multiflot d'équilibre. Comparant nos résultats avec ceux de Guélat, pour les variantes de Frank-Wolfe, on constate que le simplexe convexe converge plus vite et mieux. Ce résultat, obtenu accidentellement, est très intéressant.

Nous avons ensuite généré des problèmes de multiflot compatible à coût minimum à partir des problèmes de multiflot d'équilibre. Nous manquons d'information pour juger la qualité des solutions produites et l'algorithme s'est révélé fort instable. Il semble, cependant, qu'il puisse produire de bonnes solutions pour des problèmes comptant plusieurs milliers d'arcs et quelques dizaines de flot. Cela nous place un ordre de grandeur au-dessus des résultats publiés, mais parmi les plus petits problèmes réels.

Enfin, nous présentons aussi plusieurs idées pour une nouvelle implantation de l'algorithme, plus efficace et plus générale.

CHAPITRE PREMIER

INTRODUCTION

Le concept de multiflot généralise le concept de flot sur un réseau. La littérature anglophone parlera de "Multicommodity Network Flow", le terme "commodity" désigne un bien transporté dans le jargon du transport.

Nous allons présenter une formulation très générale pour le modèle de multiflot. Nous discuterons ensuite les spécialisations de ce modèle qui présentent un intérêt pratique. Nous terminerons ce chapitre en décrivant notre but et notre démarche.

1.1 La classe des modèles de multiflot

Soit un réseau $G = (N,A)$

où $N = \{i \in \mathbb{N} \mid 1 \leq i \leq n\}$ est l'ensemble des noeuds et

$A \subseteq \{(i,j) \in N^2 \mid i \neq j\}$, $|A| = m$ est l'ensemble des arcs.

Nous noterons aussi A, la matrice d'incidence noeud-arc de G.

Soit $K = \{i\}_{i=1}^{K}$, l'ensemble des indices des flots. Nous noterons par K, l'ensemble et sa cardinalité pour alléger la notation.

Soit $x^k = (x_a^k)_{a \in A}$, la valeur du flot k sur le réseau G.

Soit $b^k = (b_i^k)_{i \in N}$, le vecteur de demande pour la commodité k.

Soit $f: \mathbb{R}^{mxK} \rightarrow \mathbb{R}$, la fonction de coût.

Soit $g: \mathbb{R}^{mxK} \rightarrow \mathbb{R}^p$, la fonction décrivant les contraintes supplémentaires.

Le modèle général de multiflot s'écrit,

MG

$$\begin{array}{lll} \min & f((x^k)_{k \in K}) & \\ \text{s.a.} & A\, x^k = b^k & k \in K \\ & x^k \geq 0 & k \in K \\ & g((x^k)_{k \in K}) \geq 0 & \end{array}$$

Le modèle est donc caractérisé par la présence sur un réseau de plusieurs flots (variables sur les arcs satisfaisants non-négativité et conservation de flot). Ces flots se partagent les ressources du réseau et interagissent au niveau des coûts.

Bien entendu, seules certaines spécialisations présentent un intérêt pratique, reliées à des situations réelles et/ou suffisamment simples pour qu'on puisse résoudre.

1.2 Le modèle de multiflot compatible à valeur maximale

Il s'énonce,

MVM

$$\max \quad \sum_{k\in K} \alpha^k v^k$$

$$\text{s.a.} \quad A x^k = (e_{t_k} - e_{s_k})\, v^k \qquad k\in K$$

$$x^k \geq 0 \qquad k\in K$$

$$\sum_{k\in K} x^k \leq h$$

où

$\alpha^k > 0$ est le poids accordé à la commodité k.

$h \geq 0$ est le vecteur des capacités du réseau.

s_k et t_k sont respectivement la source et le puits pour le flot k.

$e_i = (\delta_{ij})_{j\in N}$, où δ_{ij} est le delta de Kronecker.

v^k est la valeur du kième flot.

Ce modèle est une généralisation du modèle de flot à valeur maximale classique.

Il est relié à l'analyse des réseaux de transport et de communication. Il mesure la charge que peut supporter un réseau donné et identifie ses points critiques. En général, ce modèle n'a pas les propriétés du flot maximal classique.

1.3 Le modèle de multiflot d'équilibre

ME

$$\min \quad \sum_{a\in A} c_a\Big(\sum_{k\in K} x_a^k\Big)$$

$$\text{s.a.} \quad Ax^k = b^k \qquad k\in K$$

$$x^k \geq 0 \qquad k\in K$$

où

c_a est une fonction convexe croissante, b^k prend la forme

$$b_i^k = \begin{cases} -\sum_{j\neq s^k} d_j^k \ , & \text{si } i=s^k \\ +\, d_i^k \ , & \text{si } i\neq s^k \\ \text{avec } d_i^k \geq 0 & i\in N - \{s^k\} \end{cases} \qquad (1.01)$$

Pour chaque k∈K, il s'agit d'acheminer ~~le flot provenant~~ d'un noeud-source s^k vers les autres noeuds, en quantité fixée.

Ce modèle apparaît naturellement dans les modèles de prévision du trafic routier. Les contraintes de flot décrivent le mouvement des usagers de leur point de départ à leur point d'arrivée. Les fonctions convexes décrivent les coûts dus à la congestion et aux distances parcourues.

Le premier principe de Wardrop stipule que puisque chaque usager cherche à minimiser ses coûts de déplacements, tous les chemins utilisés par les usagers de même origine et destination auront le même coût dans la réalité (sinon les usagers informés changeraient de chemin) et tous les chemins non-utilisés auront un plus grand coût.

La solution du modèle d'équilibre présente cette propriété. Si la fonction objective est correctement calibrée, le modèle peut simuler la circulation dans un réseau routier.

1.4 Le modèle de multiflot compatible à coût minimum

Il s'énonce,

MCM

$$\min \quad \sum_{k\in K} c^k x^k \tag{1.02}$$

$$\text{s.a.} \quad Ax^k = b^k \qquad k\in K \tag{1.03}$$

$$x^k \geq 0 \qquad k\in K \tag{1.04}$$

$$\sum_{k\in K} x^k \leq h \tag{1.05}$$

où

c^k est le vecteur des coûts pour $k \in K$.

b^k , idem à l'équation (1.01).

$h = (h_a)_{a \in A} \geq 0$ est le vecteur des capacités.

Un exemple typique est le problème de production-distribution. Il s'agit de conduire plusieurs types de biens de leurs lieux de production (source) aux points de consommation à travers un réseau de distribution ou de traitement aux capacités limitées à un coût minimal.

On retrouve aussi ce problème lors de l'étude des réseaux de communication où les flots correspondent aux messages provenant d'une même source et les capacités aux limites physiques des appareils et lignes de transmission.

1.5 Problématique

Les problèmes de multiflot sont des problèmes très difficiles. En fait, beaucoup plus difficiles que les problèmes de flot de taille comparable. Les grands succès obtenus par les algorithmes spécialisés pour les problèmes de flot dans les réseaux ne se sont pas répétés pour les modèles de multiflot,malgré les efforts des chercheurs.

Nous nous sommes intéressés aux méthodes de résolution pour le modèle de multiflot compatible à coût minimum. Ce modèle est un bon représentant de la classe des modèles de multiflot. En effet, il est simple tout en étant du même ordre de difficulté que les modèles plus complexes.

Le but de ce travail est le choix, l'implantation et l'évaluation d'un algorithme efficace pour la résolution de problèmes de grande taille.

Dans un premier temps, nous passons en revue les méthodes proposées dans la littérature. Nous présentons ensuite un algorithme qui, pour l'essentiel, suit le schéma proposé par Van der Straatten-Tilquin dans l'application de la méthode du lagrangien augmenté. Les sous-problèmes de flot à coûts convexes sont résolus à l'aide d'une adaptation de la méthode du simplexe-convexe développée par Nguyen pour le problème de multiflot d'équilibre.

Une implantation efficace a été réalisée et des résultats numériques sont présentés.

CHAPITRE DEUXIEME

L'ETAT DES RECHERCHES

Dans le présent chapitre, nous présentons une revue des méthodes de solution du modèle de multiflot compatible à coût minimum recensées dans la littérature. Dans un second temps, nous comparons les propriétés des méthodes et analysons l'expérience numérique disponible. En conclusion, nous présentons les arguments qui ont influencé le choix de l'algorithme.

2.1 Une revue des méthodes de solution

Nous allons décrire ici plusieurs des méthodes de solution qui ont été proposées ou mises en oeuvre pour solutionner le modèle MCM. Nous décrirons pour chaque méthode les simplifications et spécialisations que permettent le modèle.

Ce procédé permettra d'alléger le discours lors de l'étude comparative qui va suivre.

Il est intéressant de constater que presque tout l'arsenal des méthodes pour problèmes de grande taille sera utilisé. Plusieurs méthodes ont en effet été créees pour ce problème ou lui furent rapidement appliquées.

2.1.1 Décomposition de Dantzig-Wolfe

Cette méthode repose sur un théorème que Swoveland [22] attribue à Goldman. Ce théorème établit qu'un polytope peut s'écrire comme la somme de l'enveloppe convexe de ses points extrêmes et du cone engendré par ses rayons extrêmes.

L'application judicieuse de ce résultat nous permet de construire la formulation arc-chemin du modèle MCM. Si l'on résout ce nouveau modèle à l'aide du simplexe revisé, la détermination de la variable d'entrée correspond à la résolution d'un problème de plus court chemin. De plus, dans certaines conditions, nous obtenons une borne inférieure sur la valeur objective.

2.1.1.1 Le théorème de Goldman

Définissons les points extrêmes et les rayons extrêmes d'un ensemble convexe X.

$p \in X$ est un point extrême de X, ssi $\forall\, x_1 \in X$, $\forall\, x_2 \in X$, $\forall \theta \in [0,1]$,

$(1-\theta)\, x_1 + \theta\, x_2 = p \Rightarrow p = x_1 = x_2$

$r \in X$ est un rayon de X, ssi $\|r\| = 1$ et $\forall\, x \in X\ \forall \delta \in \mathbb{R}^+$, $x + \delta r \in X$

$r \in X$ est un rayon extrême de X, ssi $\forall\, r_1, r_2$ rayon de X, $\forall\, \theta \in [0,1]$,

$$\frac{(1-\theta) r_1 + \theta\, r_2}{\|(1-\theta) r_1 + \theta\, r_2\|} = r \Rightarrow r_1 = r_2 = r.$$

Soit $X = \{ x \mid Ax = b \quad x \geq 0\}$ un polytope, le théorème de Goldman (Swoveland [22]) nous assure que,

i) X possède un ensemble <u>fini</u> de points extrêmes et un ensemble <u>fini</u> de rayons extrêmes.

Si P indexe les points extrêmes et R les rayons extrêmes, on peut définir,

$$Y = \{(\theta,\gamma) \in \mathbb{R}^{P+R} \mid \sum_{i\in P} \theta_i = 1 \quad \theta_i \geq 0 \quad \gamma_j \geq 0 \quad \forall i\in P \quad \forall j\in R\}$$

ii) La fonction F: $Y \rightarrow X$, définie ci-après, est une bijection

$$F(\theta,\gamma) = \sum_{i\in P} \theta_i p_i + \sum_{j\in R} \gamma_j r_j \qquad (2.01)$$

Si la matrice A est la matrice d'incidence arc-chemin d'un réseau G=(N,A) et si b le vecteur de demande/disponibilité est tel que défini par l'équation (1.01) pour une source s∈N, on peut caractériser les points et rayons extrêmes de X.

a) Un point extrême p correspond à l'affectation du flot de la source s vers les destinations sur une sous-arborescence B de racine s. Notons que plusieurs arborescences peuvent correspondre au même point p si $b_i=0$ pour i∈N.

b) Un rayon (extrême) correspond à une affectation de flot sur un circuit (élémentaire).

2.1.1.2 La formulation arc-chemin du problème multiflot

Nous pouvons maintenant introduire la formulation arc-chemin du multiflot. Elle se déduit facilement du modèle MCM et de la fonction F introduite précédemment en (2.01).

Soit $X^k = \{ x^k \mid A x^k = b^k, x^k \geq 0\}$, $k \in K$.

P^k indexant les points extrêmes de X^k.

R^k indexant les rayons extrêmes de X^k.

On obtient le modèle,

$$\min \sum_{k\in K} \Big[\sum_{i\in P^k} (c^k p_i^k)\, \lambda_i^k + \sum_{j\in R^k} (c^k r_j^k)\, \gamma_j^k \Big]$$

FAC

$$\text{s.a} \sum_{k\in K} \Big[\sum_{i\in P^k} p_i^k\, \lambda_i^k + \sum_{j\in R^k} r_j^k\, \gamma_j^k \Big] \leq h$$

$$\sum_{i\in P^k} \lambda_i^k = 1 \qquad k\in K$$

$$\lambda_i^k \geq 0 \qquad i\in P^k, k\in K$$

$$\gamma_j^k \geq 0 \qquad j\in R^k, k\in K$$

Le théorème de Goldman nous assure que les modèles MCM et FAC sont équivalents et la fonction F nous permet de passer de l'un à l'autre.

Le nouveau modèle compte beaucoup moins de contraintes, soit $(m + K)$ versus $((n \times K) + m)$, mais beaucoup plus de variables, points et rayons extrêmes étant typiquement très nombreux.

2.1.1.3 Génération de colonne

Décrivons l'application du simplexe revisé à FAC. Supposons connue une base B pour le modèle FAC.

Soit (π, u), $\pi \in \mathbb{R}^m$, $u \in \mathbb{R}^K$, le vecteur des variables duales associées aux contraintes de capacité (π) et aux contraintes de convexité (u) pour la base B. Les coûts réduits se calculent alors:

i) pour un rayon extrême

$$\bar{c}_j^k = c^k r_j^k - \pi r_j^k \qquad \forall\, j \in R^k,\ \forall\, k \in K$$

$$= (c^k - \pi)\, r_j^k$$

ii) pour un point extrême

$$\bar{c}_i^k = c^k p_i^k - \pi_i^k p_i^k + u^k \qquad \forall\, i \in P^k,\ \forall\, k \in K$$

$$= (c^k - \pi)\, p_i^k + u^k$$

Pour appliquer le critère d'entrée du simplexe, il faut déterminer,

$$\min \left\{ \min_{i \in P^k} \{\bar{c}_i^k\},\ \min_{j \in R^k} \{\bar{c}_j^k\} \right\}.$$

Considérons le modèle pour $k \in K$ fixé

$SP^k(\pi)$

$$\min \quad (c^k - \pi)\, x^k$$
$$A\, x^k = b^k$$
$$x^k \geq 0$$

C'est un problème de plus court chemin et il existe de nombreux algorithmes pour sa résolution. Voir Pape [17] et Jensen-Barnes [10, chapitre 5].

Si le réseau admet, sous les coûts $(c^k - \pi)$, un circuit de longueur négative, l'algorithme proposé par Jensen et Barnes [10] ou l'application du simplexe sur le réseau détermine un circuit élémentaire de longueur négative. Construisant le rayon extrême r^k en appliquant une unité de flot sur le circuit, on obtient,

$$\bar{c}^k = (c^k - \pi)\, r^k < 0$$

et $\begin{bmatrix} r^k \\ 0 \end{bmatrix}$ est une colonne candidate pour entrer dans la base.

Si le réseau n'admet pas de circuit de longueur négative, on détermine alors une arborescence de racine s^k sur laquelle on peut circuler à coût minimal les demandes b^k. Soit p^k un tel flot, on a alors,

$$\bar{c}^k = (c^k - \pi)\, p^k + u^k = \min_{i \in p^k} \{\bar{c}^k_i\}.$$

Si $\bar{c}^k < 0$, $\begin{pmatrix} p^k \\ e_k \end{pmatrix}$ est une colonne candidate à entrer dans la base.

De plus, si $\bar{c}^k \geq 0 \quad \forall\, k \in K$, la base B est optimale.

La résolution des problèmes de plus court chemin permet donc de générer de bonnes colonnes pour entrer dans la base ou d'en prouver l'optimalité sans considérer explicitement toutes les colonnes du problème.

2.1.1.4 Generalized upper Bounding

La technique de génération de colonnes règle le problème du critère d'entrée. L'autre processus coûteux du simplexe est la gestion de la base. Une mesure de ces coûts est donnée par la taille p de la matrice à inverser explicitement. Les coûts de calcul sont de l'ordre de $O(p^3)$. La structure particulière de FAC permet d'appliquer la méthode de "Generalized upper Bounding" tel que décrite par Lasdon [12 , chapitre 3-4] et résumée en 2.1.5.

Une application judicieuse permet de limiter p au nombre de contraintes de capacités (1.05) actives. Le prix à payer est un certain nombre de produits vectoriels supplémentaires à chaque itération du simplexe.

2.1.1.5 Borne inférieure

A chaque itération, la résolution d'un problème de plus court chemin pour chaque k∈K peut fournir une borne inférieure sur la valeur optimale.

En effet, pour (π, u), si $SP^k(\pi)$ n'admet pas de circuit de longueur négative, si p^k est la solution optimale de $SP^k(\pi)$ pour tout k∈K et si Z^* est la valeur optimale de FAC alors,

$$\sum_{k\in K} (c^k - \pi)\, p^k + u\, h \leq Z^*.$$

2.1.1.6 Référence

La méthode de décomposition est attribuée à Dantzig-Wolfe [3]. On retrouve un bon exposé de la méthode dans le livre de Lasdon [12, chapitres 3-4]. La méthode a été utilisée dans le contexte du multiflot par Tomlin [23], Swoveland [22].

2.1.2 Partition primale de Rosen

Cette méthode est une spécialisation de l'approche de relaxation. Une base est choisie pour chaque flot et les contraintes de positivité des variables de base sont relaxées. Le programme linéaire résultant a

de très intéressantes propriétés. Chaque colonne est le vecteur caractéristique du cycle fondamental formé par un arc hors-base avec la base. Le modèle compte seulement une contrainte par arc. Cela explique l'intérêt de la méthode.

2.1.2.1 Partition des variables

Choisissons pour chaque $k \in K$ une base réalisable $B^k \subseteq A$. Nous noterons N^k l'ensemble des colonnes hors-base, i.e.:

$$N^k = A - B^k$$

Par B^k réalisable, nous entendons:

$$[B^k]^{-1} \; b \geq 0 \qquad k \in K$$

Sans perte de généralité, on peut permuter les colonnes du problème pour obtenir la forme,

$$\min \sum_{k \in K} c^k_{B^k} \, x^k_{B^k} + c^k_{N^k} \, x^k_{N^k}$$

$$\text{s.a.} \quad B^k x^k_{B^k} + N^k x^k_{N^k} = b^k \qquad \forall \, k \in K$$

$$x^k_{B^k}, x^k_{N^k} \geq 0 \qquad \forall \, k \in K$$

$$\sum_{k \in K} [E^k_{B^k} \, x^k_{B^k} + E^k_{N^k} \, x^k_{N^k}] \leq h$$

où $E^k_{B^k}$, $E^k_{N^k}$ correspondent aux permutations nécessaires de la matrice identité.

Relaxant les contraintes de positivité des variables de base, nous éliminons ces variables du problème.

Nous avons,

$$x^k_{B^k} = [B^k]^{-1} \; [b^k - N^k \, x^k_{N^k}]$$

Et le modèle résultant s'écrit,

$$\text{PR} \qquad \min \; \sum_{k\in K} [c^k_{N^k} - c^k_{B^k} [B^k]^{-1} N^k] \; x^k_{N^k} + \sum_{k\in K} c^k_{B^k} [B^k]^{-1} b^k$$

$$\text{s.a.} \; \sum_{k\in K} [E^k_{N^k} - E^k_{B^k} [B^k]^{-1} N^k] \; x^k_{N^k} \le h - \sum_{k\in K} E^k_{B^k} [B^k]^{-1} \; b^k$$

$$x^k_{N^k} \ge 0$$

Ce problème n'a qu'une contrainte par arc. De plus, il a une interprétation très simple en terme de flot.

La matrice A étant une matrice d'incidence noeud-arc, les bases B^k correspondent à des arbres sur le réseau (vu la nature des vecteurs b^k, on peut se limiter aux arborescences).

Toutes les quantités impliquées dans le nouveau modèle PR peuvent être calculées très économiquement si l'on utilise les structures de données appropriées. Entre autres, il n'est pas nécessaire d'inverser expli-

citement les bases B^k. On voit bien que

$c^k_{B^k} [B^k]^{-1} \; b^k$ est le coût du flot de base.

$E^k_{B^k} [B^k]^{-1} \; b^k$ est l'application du flot de base $x^k_{B^k}$ sur le réseau.

Pour un arc a hors-base pour le flot k, nous calculons,

$$[c^k_{N^k} - c^k_{B^k} [B^k]^{-1} N^k]_a = \sum_{i \in A} \delta_a^{B^k}(i) \, c^k_i$$

$$[E^k_{N^k} - E^k_{B^k} [B^k]^{-1} N^k] = \delta_a^{B^k}$$

où

$$\delta_a^B(i) = \begin{cases} \pm 1 & \text{si i appartient au cycle fondamental formé par l'arc a et la base B. Le signe dépendant de l'orientation } (\delta_a^B(a) = 1) \\ 0 & \text{sinon} \end{cases}$$

On voit donc que chaque variable du nouveau problème mesure les modifications de flot sur les cycles fondamentaux par rapport au flot de base.

2.1.2.2 Changement de base

Supposons connue une solution optimale $(\bar{x}^k_{N^k})_{k \in K}$ du problème relaxé. On calcule les valeurs des variables de base selon la formule:

$$\bar{x}^k_{B^k} = [B^k]^{-1} \; [b^k - N^k \bar{x}^k_{N^k}]$$

Si $\bar{x}^k_{B^k} \geq 0$, $\forall k \in K$ alors le vecteur $(\bar{x}^k_{B^k}, \bar{x}^k_{N^k})_{k \in K}$ est une solution optimale du problème original.

Supposons donc $\exists k \in K$ tel que $\bar{x}^k_B \not\geq 0$ et négligeons l'indice k pour alléger la notation dans ce qui suit.

Définissons les ensembles

$B^- = \{a \in B \mid \bar{x}_a < 0\}$

$N^+ = \{a \in N \mid \bar{x}_a > 0\}$

On peut montrer (Lasdon [12]) qu'il existe toujours une suite de pivots permettant d'échanger au moins une composante de B^- avec au moins une composante de N^+. Cela correspond à remplacer une contrainte violée par une contrainte inactive.

Une façon de réaliser cet échange est de calculer s tel que

$$\theta_s = \min \left[\theta_a \mid \theta_a = \frac{x^*_a}{x^*_a - \bar{x}_a}, \; a \in B^- \right]$$

où $x^*_B = [B]^{-1} b$ pour résoudre ensuite, en utilisant la base initiale B, le modèle

$$\min \; e_s x$$

$$Bx_B + N x_N = b$$

$$x \geq 0$$

$$x_i = 0 \qquad \forall i \notin BUN^+$$

Si s est de base à l'optimum, il en est sorti par un pivot dégénéré.

2.1.2.3 Changement de base sur le réseau

Sur le réseau, dans notre contexte, on peut suggérer un mécanisme qui évite le recours à la résolution de problèmes de flot à coût minimum et qui permet de passer d'arborescences en arborescences.

ETAPE 1 - Recherche de l'arc sortant

Parcourir l'arborescence initiale B des feuilles vers la racine (à l'aide de la fonction de traverse en préordre inverse) jusqu'au premier arc a tel que $\bar{x}_a < 0$.

Soit $q = (i, j)$ cet arc.

ETAPE 2 - Recherche de l'arc entrant

Choisir

$p \in C = \{(\ell, j) \mid \bar{x}_{\ell j} > 0 \text{ et } (\ell, j) \in A\}$

ETAPE 3 - Pivot

Faire $B \leftarrow B - \{q\} + \{p\}$

FIN

L'ensemble C est non-vide par définition de p, par les équations de conservation de flot et la définition du flot $\bar{x}$, en effet...

$$b_j \geq 0 \qquad \forall\, j \notin S$$

$$\bar{x}_p < 0$$

$$\bar{x}_a \geq 0 \qquad \forall\, a \in A - B$$

D'où

$$\sum_{\substack{j(a)=j(p) \\ a\neq p}} \bar{x}_a = \sum_{i(a)=j(p)} \bar{x}_a + b_{j(p)} - x_p > 0$$

D'où

$\exists\, q \in N$ tel que $\bar{x}_q > 0$, C.Q.F.D.

2.1.2.4 L'algorithme

Après le développement qui précède, l'algorithme s'énonce simplement comme une spécialisation de l'approche de relaxation.

ETAPE 0 - Initialisation

Choisir des bases B^k (arbres ou arborescences) pour $k \in K$.

ETAPE 1 - Résolution du programme-maître

Résoudre PR pour les bases B^k, $k \in K$ obtenant $(\bar{x}^k_{N^k})_{k \in K}$ optimal.

ETAPE 2 - Critère d'arrêt

Si $\bar{x}^k_{B^k} \geq 0 \quad \forall\, k \in K$ on a terminé sinon on va à l'étape 3.

ETAPE 3 - Changement de base

Pour au moins un $k \in K$ tel que $\bar{x}^k_{B^k} \not\geq 0$, procéder à un changement de base selon l'un ou l'autre des procédés décrits.

On voit facilement l'importance de "bonnes" bases de départ pour limiter le nombre de problèmes à résoudre à l'étape 1. Il faut aussi bien choisir les nouvelles bases à l'étape 3 pour limiter le nombre d'itérations.

2.1.2.5 Références

La méthode de partition primale est généralement attribuée à Rosen [20]. La spécialisation au problème de multiflot compatible a été proposée par Grigoriadis et White [6] et par Kennington et Shalaby [11].

2.1.3 Allocation de ressource

Si nous allouons à chaque flot une partie de la capacité totale, nous pouvons calculer un multiflot optimal (pour cette allocation) en résolvant un problème de flot à coût minimum pour chaque $k \in K$. Nous pouvons donc déterminer une solution optimale pour le modèle MCM si nous choisissons une allocation optimale de la capacité.

La méthode d'allocation de ressource résout le modèle MCM en calculant une telle allocation optimale.

2.1.3.1 Modèle d'allocation

Allouons donc à chaque flot $k \in K$ une partie y^k de la capacité totale h, nous obtenons le modèle

$$\min \quad \sum_{k \in K} v^k(y^k)$$

$$\text{s.a.} \quad \sum_{k \in K} y^k = h$$

$$y^k \geq 0 \qquad k \in K$$

où

$$SPP^k(y) \qquad v^k(y) = \min \; c^k x + Mt$$

$$\text{s.a. } Ax + t = b^k$$

$$0 \leq x \leq y$$

$$t \geq 0$$

Les variables artificielles $(t_i)_{i \in N}$ permettent d'affecter de grands coûts aux allocations irréalisables en prenant $(M_i)_{i \in N}$ arbitrairement grand.

Pour $y = (y^k)_{k\in K}$ fixé $v^k(y^k)$ se calcule en résolvant un problème de flot à coût minimum pour chaque $k\in K$. On pense donc au simplexe sur le réseau ou à la méthode des arcs non-conformes (out-of-kilter) pour la résolution.

Le dual du sous-problème $SPP^k(y)$ s'écrit

$$SPD^k(y) \qquad \max \quad \Pi\, b^k + \mu\, y$$

$$\text{s.a.} \quad \Pi\, A + \mu \geq c^k$$

$$\Pi \leq M$$

$$\mu \leq 0$$

A la vue du modèle $SPD^k(y)$, on s'aperçoit que chaque fonction v^k est l'enveloppe supérieure d'une collection finie de fonctions affines. Par conséquent, v^k est convexe, continue, linéaire par morceaux.

La fonction $v((y^k)_{k\in K}) = \sum_{k\in K} v^k(y^k)$ est donc convexe, continue, linéaire par morceaux.

2.1.3.2 Sous-gradient

Soit (Π^k, μ^k) solution optimale de SPD^k pour chaque $k\in K$. Le vecteur $\mu = (\mu^k)_{k\in K}$ est un sous-gradient de v en $y = (y^k)_{k\in K}$, c'est-à-dire

$$v(y) + \mu (y_* - y) = \sum_{k\in K} [\pi^k b^k + \mu^k y^k] + \sum_{k\in K} [\mu^k (y_*^k - y^k)]$$

$$= \sum_{k\in K} \pi^k b^k + \mu^k y_*^k$$

$$\leq v(y_*)$$

D'où

$$\forall\, y \geq 0 \quad , \quad \forall\, y_* \geq 0 \ ,$$

$$v(y) + \mu (y_* - y) \leq v(y_*)$$

C.Q.F.D.

2.1.3.3 Linéarisation externe

Soit L^k l'ensemble fini des solutions de base distinctes du sous-problème $SPD^k(y)$ indexé par $\mathbb{L}^k$

$$L^k = \{(\pi_i^k, \mu_i^k)\}_{i\in\mathbb{L}^k} = \{(\pi^k, \mu^k) \mid (\pi^k, \mu^k) \text{ sol. de base}\}.$$

Soit L, l'ensemble produit des L^k indexé par $\mathbb{L}$

$$L = \{(\pi, \mu)_i\}_{i\in\mathbb{L}} = \prod_{k\in K} L_K \ .$$

Calculons

$$v(y) = \sum_{k\in K} \max \{\pi b^k + \mu y^k \mid \pi A + \mu \geq c^k,\ \pi \leq M,\ \mu \leq 0\}$$

$$= \sum_{k\in K} \max_{i\in L^k} \{\pi_i\, b^k + \mu_i\, y^k\}$$

$$= \max_{i\in L} \{ \sum_{k\in K} \pi_i\, b^k + \mu_i\, y^k\}$$

On peut maintenant réécrire le programme d'allocation sous la forme:

$$\min \quad \sigma$$

$$\text{s.a.} \quad \sum_{k\in K} \pi_i\, b^k + \mu_i\, y^k \leq \sigma \qquad i\in L$$

$$\sum_{k\in K} y^k = h$$

$$y \geq 0$$

Ce problème compte un grand nombre de contraintes. Cependant, seules les contraintes décrivant la fonction v au voisinage d'un point y sont actives.

Nous allons décrire une façon de construire l'ensemble des contraintes actives au voisinage de l'optimum,que l'on peut interpréter comme une approche de relaxation.

ETAPE 0 - Initialisation

Poser $j = 0$

ETAPE 1 - Résolution du programme relaxé

Résoudre $\min \ \sigma$

$$\text{s.a.} \quad \Sigma \ \pi_i \ b^k + \mu_i \ y^k \leq \sigma \qquad 1 \leq i \leq j$$

$$\sum_{k \in K} y^k = h$$

$$y^k \geq 0 \qquad k \in K \ .$$

Faire $j = j + 1$.

Notons y_j, σ_j la solution optimale.

ETAPE 2 - Résolution des sous-problèmes

Résoudre $SPD^k(y_j^k)$ obtenant π_j^k, μ_j^k les solutions optimales.

ETAPE 3 - Test d'optimalité

Si $\sum_{k \in K} \pi_j^k \ b^k + \mu_i^k \ y_j^k = \sigma_j$

alors on a terminé et y_j est optimal sinon on va à l'étape 1.

FIN

On peut négliger les valeurs des π_i^k si l'on note que

$$\sum_{k \in K} \pi_i^k \ b^k + \mu_i^k \ y^k = \sum_{k \in K} \pi_i^k \ b^k + \mu_i^k \ y_i^k + \sum_{k \in K} \mu_i^k \ y^k - \mu_i^k \ y_i^k$$

$$= v(y_i) + \mu_i (y - y_i)$$

Cette transformation met en évidence la relation entre les sous-gradients et la linéarisation de la fonction.

2.1.3.4 Méthode du sous-gradient

Soit le modèle

$$\min \quad v(\lambda)$$
$$\lambda \in X$$

où

$v: \mathbb{R}^n \rightarrow \mathbb{R}$ sous-différentiable sur X

$X \subseteq \mathbb{R}^n$

La méthode du sous-gradient s'applique à l'optimisation de fonctions sous-différentiables. Elle génère une suite de points de façon itérative en utilisant l'information fournie par un sous-gradient au point courant.

On calcule récursivement

$$\lambda^{n+1} = P(\lambda^n - p^n\, d(\lambda^n))$$

où

$P: \mathbb{R}^n \rightarrow X$ une fonction de projection

$p^n \in \mathbb{R}^+$ un pas scalaire

$d(\lambda)$ un sous-gradient de v en λ.

La fonction P est, par exemple, une projection à distance minimale.

Cet algorithme est convergent pour toutes les séquences de pas satisfaisant

$$p_n = r_n \, (v(\lambda_n) - v^*) \; / \; \| \; d(\lambda_n) \; \|^2$$

où

v^* est la valeur optimale

r_n une suite telle que $r_n \geq 0$

$$r_n \to 0$$

$$\sum_{i=0}^{n} r_i \to \infty$$

En général, on ne connaît pas v^* la valeur optimale et on la remplace par une borne inférieure.

Ce pas heuristique est préféré à la recherche du pas optimal à cause du coût élevé d'évaluation de la fonction v et de la projection P.

Remarquons qu'un tel pas n'assure pas la décroissance de l'objectif.

On cesse généralement l'algorithme lorsque

$$\left| \frac{v(\lambda_n) - v^*}{v^*} \right| < \varepsilon$$

où ε est la précision cherchée. On voit l'importance prise par la connaissance d'une bonne borne inférieure.

2.1.3.5 Projection

Pour terminer, il faut décrire la fonction de projection P. Il est naturel de projeter à distance minimale.

Soit $z = \lambda_n - p_n\, d(\lambda_n)$ on définit P comme

P(z) est solution optimale de

$$\min \quad \frac{1}{2} \sum_{a\in A} \sum_{k\in K} (y_a^k - z_a^k)^2$$

$$\text{s.a.} \quad \sum_{k\in K} y^k = h$$

$$y^k \geq 0 \,, \quad k\in K$$

2.1.3.6 Résolution du programme de projection

Ce problème est séparable par arc. Chacun de ces problèmes peut être résolu analytiquement (de façon exacte). Chaque problème a la forme,

$$\min \quad \frac{1}{2} \sum_{j=1}^{n} (x_j - z_j)^2$$

$$\sum_{j=1}^{n} x_j = h$$

$$x_j \geq 0 \quad \forall\, j,\ 1 \leq j \leq n$$

où $z\in\mathbb{R}^n$ est fixé et n désigne ici la taille du problème h est un scalaire.

Les conditions de K - T de ce problème s'écrivent

$x^* \in \mathbb{R}^n$ optimal ssi $\exists\, v^* \in \mathbb{R}^n$, $\lambda^* \in \mathbb{R}$

tel que $\forall\, j,\ 1 \le j \le n$

$$x_j^* - z_j = v_j^* - \lambda^* \tag{2.02}$$

$$v_j \ge 0 \tag{2.03}$$

$$x_j^* \ge 0 \tag{2.04}$$

$$\sum_{j=1}^{n} x_j^* = h \tag{2.05}$$

$$x_j\, v_j = 0 \tag{2.06}$$

Définissons

$$x_j(\lambda) = \max_{1 \le j \le n} \{z_j - \lambda,\ 0\}$$

$$v_j(\lambda) = \max_{1 \le j \le n} \{-z_j + \lambda,\ 0\}$$

Clairement, le couple $x(\lambda)$, $v(\lambda)$ satisfait (2.02-04) et (2.06) pour tout λ. Il faut donc chercher λ^* tel que

$$\sum_{j=1}^{n} x_j(\lambda^*) = h$$

Sans perte de généralité, supposons que

$$z_1 \le z_2 \ldots \le z_n$$

sons

$$h(\lambda) = \sum_{j=1}^{n} x_j(\lambda) = \sum_{j=1}^{n} \max \{z_j - \lambda, 0\}$$

$$= \sum_{j=J_\lambda+1}^{n} z_j - [n - J_\lambda] \lambda$$

où

$$J_\lambda = \max_{1 \le j \le n} \{J \mid z_J - \lambda < 0\}.$$

De plus, $h(\lambda)$ est une fonction décroissante et convexe, car c'est une somme de fonctions décroissantes et convexes.

Posons

$$J^* = \max_{1 \le j \le n} \{J \mid \sum_{j=J+1}^{n} z_j - (n - J) z_J \ge h\}$$

et

$$\lambda^* = \frac{-h + \sum_{j=J^*+1}^{n} z_j}{n - J^*}$$

On a alors

$$z_J - \lambda^* = \frac{h + [\sum_{j=J^*+1}^{n} z_j - (n - J^*) z_J]}{n - j^*}$$

Par définition de λ^*,

$$\forall\, j \le J^* \qquad z_j - \lambda^* \le 0$$

$$\forall\, j > J^* \qquad z_j - \lambda^* > 0$$

On a donc pour conclure,

$$h(\lambda^*) = \sum_{j=1}^{n} x(\lambda^*) = \sum_{j=1}^{n} \max (z_j - \lambda^*, 0)$$

$$= \sum_{j=J^*+1}^{n} z_j - \lambda^*$$

$$= \sum_{j=J^*+1}^{n} z_j - \left[\frac{h + \sum_{j=J^*+1}^{n} z_j}{n - J^*} \right] (n - J^*)$$

$$= h$$

La résolution du sous-problème correspond donc à calculer successivement J^*, λ^* et $x(\lambda^*)$. Toute la difficulté de la résolution est dans la nécessité d'ordonner les valeurs z_j. Cela nécessite ou équivaut à un tri, processus coûteux.

2.1.3.7 Références

Le modèle d'allocation a été proposé par Robacker [19], Sakarovitch [21], Swoveland [22], Held-Wolfe-Crowder [9], Kennington et Shalaby [11].

Swoveland [22] a expérimenté la méthode de linéarisation externe. Held-Wolfe-Crowder [9] utilisent la méthode des sous-gradients pour résoudre le modèle MVM. Nous leurs devons la méthode de projection de la sec-

tion 2.1.3.5. Kennington-Shalaby ont généralisé les travaux de Held-Wolfe-Crowder au modèle MCM.

Les résultats théoriques concernant la méthode de sous-gradient sont attribués à Polyak [18].

2.1.4 Résolution du dual par la méthode de Bender

L'efficacité de la méthode de sous-gradient pour résoudre le modèle d'allocation de ressource repose sur la connaissance d'une bonne borne supérieure (voir section 2.1.3.4).

Considérons le dual du modèle MCM. Si nous fixons la valeur des variables duales associées aux arcs et résolvons le problème résultant, nous obtenons une borne supérieure sur la valeur optimale de MCM. Cette borne est obtenue à un coût raisonnable en calculant un problème de plus court chemin pour chaque flot.

2.1.4.1 Le modèle dual

Construisons le dual du modèle MCM.

Pour simplifier la présentation, ajoutons au modèle MCM les contraintes redondantes

$$x^k \leq M \qquad \forall\, k \in K$$

Pour $h << M$.

Dès lors, le dual s'écrit:

$$D \qquad \max \sum_{k\in K} [\pi^k b^k - e^k M] - \lambda h$$

$$\pi^k A - e^k - \lambda \geq c^k \qquad \forall\, k\in K$$

$$e^k \geq 0 \qquad \forall\, k\in K$$

$$\lambda \geq 0$$

Remarquons que pour λ fixé, le modèle est séparable. Posons donc,

$$SPD^k(\lambda) \qquad v^k(\lambda) = \max \{\pi b^k - e M \mid \pi A - e M - \lambda \geq c^k,\ e \geq 0\}$$

Et, par dualité...

$$SPP^k(\lambda) \qquad v^k(\lambda) = \min \{(c^k+\lambda) x \mid Ax = b^k,\ 0 \leq x \leq M\}$$

Si le modèle original MCM est réalisable, les fonctions v^k sont bien définies, car $SPP^k(\lambda)$ est réalisable et bornée pour $k\in K$.

De plus, si le réseau n'admet pas de cycle négatif (par exemple, si $c^k \geq 0$) et si M est choisi suffisamment grand, il s'agit de problèmes de plus court chemin.

La fonction

$$v(\lambda) = \sum_{k \in K} v^k(\lambda) - \lambda h$$

est concave, linéaire par morceaux, car somme de telles fonctions. De fait, chaque fonction v^k est l'enveloppe inférieure d'une quantité finie de fonctions affines.

Les constructions précédentes nous permettent d'écrire le modèle

PDR $\quad \max \quad v(\lambda)$

$$\lambda \geq 0$$

Par dualité linéaire, on peut montrer que ce modèle a même valeur optimale et mêmes solutions optimales en λ que le modèle D.

On obtient aussi ce modèle en prenant la dualisation partielle des contraintes de capacité.

2.1.4.2 Sous-gradient

Soit $\bar{x}^k$ sol. opt. de $SPP^k(\lambda)$

Soit $d = h - \sum_{k \in K} \bar{x}^k$

alors

-d est un sous-gradient en λ de - v.

PREUVE

$$v(\lambda) + (\lambda^1 - \lambda)\, d = \sum_{k\in K} [(c^k + \lambda)\, \bar{x}^k] - \lambda h + (\lambda - \lambda^1)(h - \sum_{k\in K} \bar{x}^K)]$$

$$= \sum_{k\in K} (c^k + \lambda^1)\, \bar{x}^k \; - \; \lambda^1 h$$

$$\geq \sum_{k\in K} \min \{(c^k + \lambda^1)\, x \mid Ax = b,\ 0 \leq x \leq M\} \; - \lambda^1 h$$

$$= \sum_{k\in K} v^k(\lambda^1) \; - \; \lambda^1 h$$

$$= v(\lambda^1)$$

D'où

$$v(\lambda) + (\lambda^1 - \lambda)\, d \geq v(\lambda^1),$$

C.Q.F.D.

2.1.4.3 Méthode de résolution

Le modèle PDR présente des propriétés analogues au modèle AR à ceci près que nous voulons maximiser une fonction concave (voir section 2.1.3).

On peut envisager de résoudre exactement PDR par linéarisation externe de la fonction v,comme pour le modèle d'allocation. Cependant, cette technique pouvant s'interpréter comme une variante de la méthode Dantzig-Wolfe, nous n'en présentons pas les détails.

Nous allons décrire brièvement l'interprétation de la méthode du sous-gradient que nous avons décrite en 2.1.3.4.

2.1.4.4 Méthode du sous-gradient

Il faut décrire une règle pour la suite $\lambda_n \geq 0$. On prendra

$$\lambda_{n+1} = P(\lambda_n + p_n \, d\,(\lambda_n))$$

où

$$p_n = r_n \cdot \frac{v^* - v(\lambda_n)}{\| d(\lambda_n) \|^2}$$

avec r_n tel que

$$\left.\begin{array}{l} r_n \to 0 \\ \sum_{i=0}^{n} r_i \to \infty \\ r_n \geq 0 \end{array}\right\} \quad \forall n \in \mathbb{N}$$

v^* est la valeur optimale ou une borne supérieure sur la valeur optimale.

$-d(\lambda)$ est un sous-gradient en λ de $-v$. Nous savons comment construire un tel sous-gradient.

$P : \mathbb{R}^m \rightarrow \mathbb{R}^{m+}$ fonction de projection a distance minimale sur l'ensemble réalisable. Dans le cas du modèle PR, cette projection est triviale et s'écrit

$$P_a(\lambda) = \max \{0, \lambda_a\} \qquad \text{pour } a \in A.$$

2.1.4.5 Référence

La méthode présentée ici est proposée par Gondran-Minoux [4]. On peut aussi refaire tout le raisonnement en utilisant la théorie de la relaxation lagrangienne.

2.1.5 Generalized upper Bounding

L'application directe du simplexe revisé pour résoudre le modèle MCM (linéaire) implique l'inversion et le maintien d'une base dont la dimension est de l'ordre de $0 (n \times K + m)$. Les besoins en mémoire pour stocker l'inverse sont de l'ordre du carré de la taille de la base. Le nombre d'opérations nécessité par l'inversion est aussi de l'ordre du cube.

Nous allons décrire une méthode de gestion de la base qui exploite la structure de la matrice de contrainte et la structure de réseau sous-jacente. Son principal intérêt est de limiter à o(m) la taille de la matrice à inverser explicitement. Il s'agit d'un gain important. Le prix à payer se limite à quelques produits scalaires et un pivot plus complexe.

2.1.5.1 Structure de la matrice de contraintes

Nous devons modifier légèrement le modèle MCM pour lui donner la forme standard. Notons d'abord que la matrice d'incidence noeud-arc est de rang $r < n$. Ce problème est réglé en retirant autant de lignes que nécessaire.

On ajoute aussi des variables d'écarts $(t_a)_{a \in A}$, positives de coûts nuls. De plus, un artifice classique, la fonction objective est ajouté au modèle pour donner le tableau suivant:

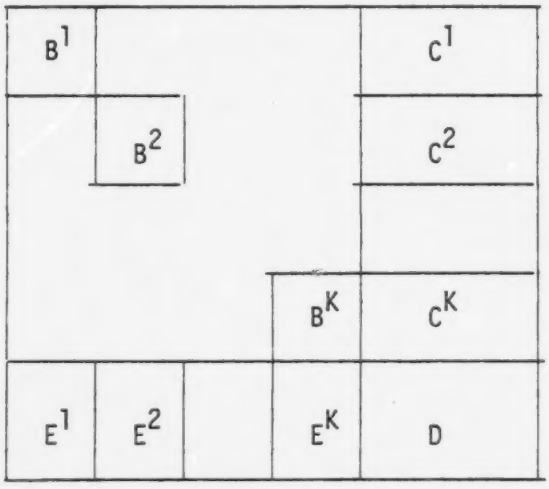
min z

$$
\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}
\multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} \\
\multicolumn{2}{|c|}{A} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} \\
\multicolumn{2}{c}{} & \multicolumn{2}{|c|}{A} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} \\
\multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{|c|}{A} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} \\
\multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} \\
\hline
I_m & I_m & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} \\
\hline
c^1 & c^2 & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} & \multicolumn{2}{c}{} \\
\end{array}
\quad \cdots \quad
\begin{array}{|c|c|c|c|}
\multicolumn{2}{|c|}{A} & \multicolumn{2}{c}{} \\
\hline
I_m & I_m & \multicolumn{2}{c}{} \\
\hline
c^K & 0 & -1 & \\
\end{array}
\qquad
\begin{array}{|c|}
x^1 \\ \hline x^2 \\ \hline \vdots \\ \hline x^K \\ \hline t \\ \hline z \\
\end{array}
\quad = \quad
\begin{array}{|c|}
b^1 \\ \hline b^2 \\ \hline \vdots \\ \hline b^K \\ \hline h \\ \hline 0 \\
\end{array}
$$

$x^k \geq 0 \quad \forall\, k \in K, \quad t \geq 0, \quad z$ libre

2.1.5.2 Structure de la base et partitionnement

Considérons une base B du programme présenté préalablement ordonnée pour exhiber la structure suivante:



Chaque B^k, matrice r x r est une base de A. Le fait que $\mathbb{B}$ soit de plein rang garantit l'existence des bases B^k. Les m+1 dernières colonnes correspondent aux variables de base résiduelles pour chaque bloc, aux variables d'écart et la variable z, cette dernière étant toujours de base. Les matrices E^k et D correspondent aux (m+1).dernières contraintes du modèle.

La matrice $\mathbb{B}$ a donc rang $r_{\mathbb{B}}$, $r_{\mathbb{B}} = (r \times K) + m + 1$.

Nous conviendrons que la colonne associée à z est la dernière colonne de $\mathbb{B}$.

Nous introduisons maintenant la matrice T de dimension et rang $r_{|B}$

$$T = \begin{array}{|c|c|} \hline I_{rxK} & V \\ \hline 0 & I_{m+1} \\ \hline \end{array}$$

où

$$V = - \begin{array}{|c|c|c|} \hline [B^1]^{-1} & & \\ \hline & [B^2]^{-1} & \\ \hline & & [B^k]^{-1} \\ \hline \end{array} \times \begin{array}{|c|} \hline C^1 \\ \hline C^2 \\ \hline \\ \hline C^k \\ \hline \end{array}$$

post-multipliant $|B$ par T, nous obtenons une matrice triangulaire inférieure par bloc

$$|BT = \begin{array}{|c|c|c|c|c|} \hline B^1 & & & & \\ \hline & B^2 & & & 0 \\ \hline & & & & \\ \hline & & & B^K & \\ \hline E^1 & E^2 & & E^K & B \\ \hline \end{array}$$

où

$$B = D - \sum_{k \in K} E^k [B^k]^{-1} C^k$$

Puisque $\mathbb{B}$ et T sont de plein rang, $\mathbb{B}T$ l'est aussi. Par conséquent, B est de plein rang.

2.1.5.3 Résolution des systèmes d'équations

Une bonne part du travail impliqué par l'application du simplexe revisé consiste à résoudre les systèmes

$$\pi \quad \mathbb{B} = e_2 , \quad \mathbb{B}\, x_{\mathbb{B}} = b \qquad (2.07)\ (2.08)$$

où

$$\pi = ((\pi_{B^k})_{k \in K}, \pi_D) , \quad x_{\mathbb{B}} = ((x^k_{B^k})_{k \in K} , x_D)$$

$$e_2 = ((0_r)_{k \in K}, e_{m+1}) , \quad b = ((b^k)_{k \in K} , h, 0)$$

On peut éviter l'inversion de $\mathbb{B}$ en exploitant les constructions introduite en 2.1.5.2. Post-multiplions les deux côtes de (2.07) par T, on obtient

$$\pi\, \mathbb{B}T = e_2\, T = e_2$$

Et puisque $\mathbb{B}T$ est triangulaire inférieure par bloc, on calcule successivement

$$\pi_D = e_{m+1}\, B^{-1}$$

$$\pi_{B^k} = \pi_D\, E^k [B^k]^{-1} , \quad \forall\, k \in K$$

Pour calculer x_{iB} , on résout d'abord

$$B^T \; z_B = b$$

où

$$z = ((z_{B^k})_{k \in K} \; , \; z_D)$$

en calculant successivement

$$z_{B^k} = [B^k]^{-1} \; b^k$$

$$z_D = B^{-1} [h - \sum_{k \in K} E^k \; z_{B^k}]$$

Finalement, on pose

$$x_{iB} = T \; z_B$$

d'où on déduit,

$$x_D = [B^{-1} \; h - \sum_{k \in K} E^k \; [B^k]^{-1} \; b^k]$$

$$x_{B^k} = [B^k]^{-1} [b^k - C^k \; x_D]$$

On voit donc que pour résoudre les systèmes (2.07) et (2.08), il suffit d'inverser les matrices B et B^k pour $k \in K$

2.1.5.4 Méthode de pivotage

Il faut développer des règles de pivotage complexe pour conserver la structure de la base d'une itération à l'autre. Il faut maintenir les matrices

$$B^{-1} \text{ et } [B^k]^{-1} \; , \; E^k \; , \; C^k \text{ pour } k \in K \; .$$

Soit E, matrice $r_B \times r_B$ de plein rang. Considérons ${}^*B^{-1} = E\ B^{-1}$ et ${}^*T$, la matrice de transformation associée. Nous nous intéressons à la relation entre ${}^*B^{-1}$ et B^{-1}. Observons d'abord

$$
\begin{aligned}
[{}^*B\,{}^*T]^{-1} &= {}^*T^{-1}\,{}^*B^{-1} \\
&= {}^*T^{-1}\ E\ B^{-1} \\
&= {}^*T^{-1}\ E\ T\ T^{-1}\ B^{-1} \\
&= {}^*T^{-1}\ E\ T\,[B\,T]^{-1}
\end{aligned}
$$

Et sous forme partitionnée, on obtient

$$
\begin{array}{|c|c|}\hline — & — \\ \hline — & {}^*B^{-1} \\ \hline \end{array}
=
\begin{array}{|c|c|}\hline I & -V^* \\ \hline 0 & I \\ \hline \end{array}
\times
\begin{array}{|c|c|}\hline E_1 & E_2 \\ \hline E_3 & E_4 \\ \hline \end{array}
\times
\begin{array}{|c|c|}\hline I & V \\ \hline 0 & I \\ \hline \end{array}
\begin{array}{|c|c|}\hline — & — \\ \hline — & B^{-1} \\ \hline \end{array}
$$

D'où

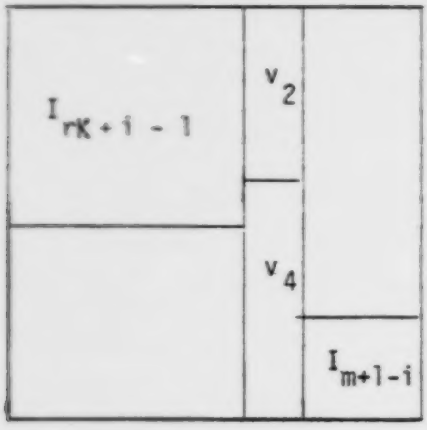
$$^{*}B^{-1} = (E_3 V + E_4)\, B^{-1}$$

La construction précédente nous permet de développer les règles de pivotage. Nous distinguons deux cas:

CAS 1) La variable sortante fait partie de l'ensemble D, soit

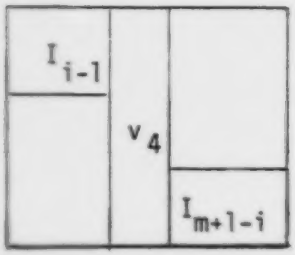
x_{Di}

Le pivot est réalisé ici de la même façon que pour la méthode du simplexe revisé. La matrice $|B^{-1}$ est pré-multipliée par une matrice E, élémentaire-colonne déduite de la colonne entrante et de la ligne où se produit le pivot. E est de plein rang et s'écrit



On en déduit que

$$^{*}B^{-1} = \quad B^{-1}$$



Il suffit, pour compléter le pivot, de mettre à jour les matrices C^k remplaçant la colonne de ces matrices par les valeurs correspondant à la variable entrante.

Les autres quantités demeurant inchangées, nous pourrons poursuivre les calculs avec la prochaine itération du simplexe.

Cas 2) La colonne sortante n'est pas partie de l'ensemble D, disons $x^k_{B^k_i}$.

La façon la plus économique de réaliser le pivot consiste à échanger d'abord la colonne sortante avec une colonne de l'ensemble D. Examinons les conditions qui permettent un tel échange. Considérons la matrice suivante, sous-matrice de B

B^k	C^k

Pré-multipliant par $[B^k]^{-1}$, nous obtenons

I	$[B^k]^{-1}\ C^k$

Désignons par v la ligne i de la partie droite, soit

$$v = [[B^k]^{-1}\ C^k]_{i.} = [B^k]^{-1}_{i.}\ C^k$$

Une condition nécessaire et suffisante pour l'existence d'une colonne j, candidate pour la permutation, est simplement que

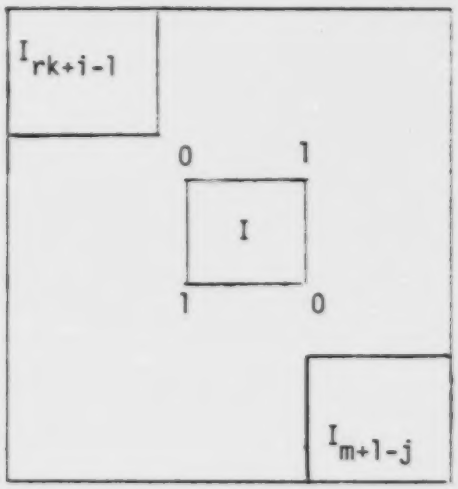
$$v \neq 0$$

Toute colonne D_j, telle que $v \neq 0$, est candidate à la permutation Notons que cela n'est possible que si la colonne D_j appartient au bloc k.

Considérons les 2 sous-cas suivants:

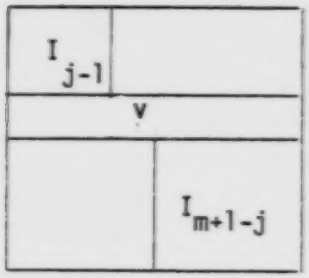
Cas 2a) La permutation est possible avec la colonne $D_j (v_j \neq 0)$
La permutation des colonnes B_i^k et D_j correspond à la matrice E telle que

$E =$



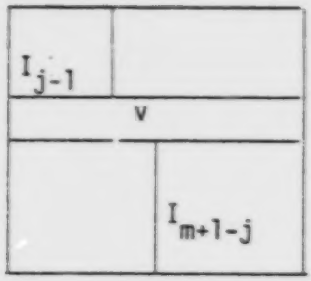
D'où

$E_3 V + E_4 =$



Et nous avons

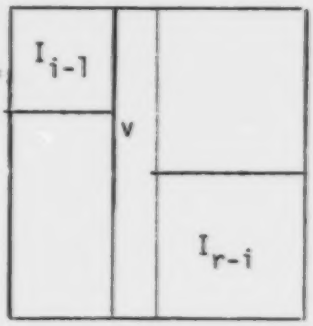
$^*B^{-1} =$



B^{-1}

On termine la mise à jour de la base en revisant la colonne $C^k_{.j}$. de C^k et la colonne $E^k_{B^k_i}$ de E^k. La base B^k a été modifiée par un pivot et l'inverse est mis à jour en la pré-multipliant par une matrice élémentaire-colonne

$[^*B^k]^{-1} =$



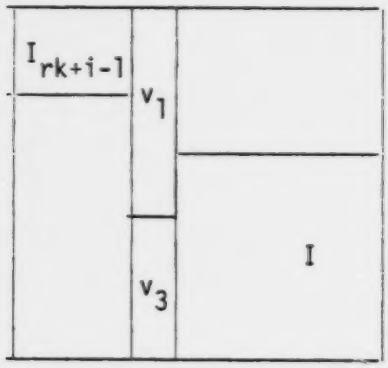
$[B^k]^{-1}$

La permutation complétée, nous retrouvons le cas 1 et nous procédons comme indiqué.

Cas 2b) On ne peut effectuer la permutation ($v = 0$).

De façon similaire au cas 1, l'inverse est mis à jour en pré-multipliant $_{B}^{-1}$ pour une matrice élémentaire-colonne E.

$E =$



D'où

$$E_3 V + E_4 = I_{m+1} \qquad \text{car } v = 0$$

Et finalement, nous avons

$$^{*}_{B}{}^{-1} = {}_{B}{}^{-1}$$

Le pivot est complété en mettant à jour E^k, en remplaçant la colonne pertinente. La base B^k est modifiée par un pivot et son inverse mis à jour.

Ceci termine la description des règles de pivotage. A chaque itération, le nouvel inverse ${}^{*}B^{-1}$ est déterminé en pré-multipliant B^{-1} par au plus une matrice élémentaire-colonne et une matrice élémentaire-ligne. Au plus une des inverses $[B^k]^{-1}$ est mis à jour par un pivot. Le reste du travail se limite à remplacer certaines colonnes des matrices E^k, C^k.

2.1.5.5 Structure de réseau

On peut exploiter la structure de réseau pour alléger la structure de donnée et l'effort de résolution.

D'abord, on note que chaque colonne correspondant à un arc n'a que quatre éléments non-nuls décrivant l'arc et son coût. Les matrices E^k, B^k et C^k peuvent être représentées de façon très compactes.

De plus, de nombreuses équations à la section 2.1.5.3 impliquent les matrices $[B^k]^{-1}$. Typiquement, il faut calculer

$$c[B^k]^{-1} \qquad \text{et} \qquad [B^k]^{-1} b \qquad (2.09)\ (2.10)$$

Ces quantités peuvent être calculées directement sur le réseau. En effet, à chaque base B^k correspond une forêt sur le réseau (un arbre, si le réseau est connexe). La structure de donnée appropriée permet de calculer (2.09) et (2.10) en $O(n)$ opérations. De plus, l'espace nécessaire pour conserver cette structure est de $O(n)$, plutôt que $O(n^2)$, pour l'inverse explicite.

De plus, cette structure permet d'effectuer les pivots sur les bases B^k exigées par le cas 2 de la règle de pivotage (2.1.5.4) en O(n) opérations.

On réalise ici un gain d'un ordre de grandeur sur le temps et l'espace nécessaire à la gestion des bases B^k. La structure et la méthode de pivot sont décrites en détail au chapitre 3.

2.1.5.6 Référence

Les résultats des sections 2.1.5.2-4 sont dus à Hartman. On en trouve un excellent exposé dans Lasdon [12]. Les règles de pivotage sont légèrement différentes de celles exposées dans Lasdon.

2.1.6 La méthode du lagrangien augmenté

Cette approche est par essence duale, car elle fournit une suite de valeurs non-réalisables.

Nous nous appuyons sur les travaux de Bertsekas [2], sur la méthode du lagrangien augmenté, et l'interprétation de Van der Straatten-Tilquin [24], pour le modèle MCM.

La méthode du lagrangien augmenté combine méthode de pénalité et relaxation lagrangienne. On peut la voir comme l'application de la relaxation lagrangienne à un modèle dont la fonction objectif a été augmentée d'un terme de pénalité.

Les contraintes d'inégalité sont transformées en contrainte d'égalité, en ajoutant des variables d'écart quadratiques. Ces variables sont éliminées du modèle résultant, car on peut déterminer analytiquement leurs valeurs.

2.1.6.1 Modélisation

Pour le modèle MCM, cela donne lieu aux constructions suivantes:

Considérons pour $u > 0$ et $\lambda = (\lambda_a)_{a \in A}$ la fonction

$$L_u(x, \lambda) = \left[\sum_{k \in K} c^k x^k \right] + \frac{1}{2u} \sum_{a \in A} \left[\max^2\{0, \lambda_a + u\left(\sum_{k \in K} x_a^k - h \right)\} - \lambda_a^2 \right]$$

On peut écrire les modèles

$$PL(\lambda) \qquad L_u(\lambda) = \min L_u(x, \lambda)$$

$$\text{s.a.} \quad Ax^k = b^k \qquad k \in K$$

$$0 \leq x^k \leq M \qquad k \in K$$

et

$$PL \qquad L^* = \max_{\lambda \geq 0} L_u(\lambda)$$

Les bornes supérieures M sont choisies arbitrairement grandes de telle sorte qu'elles sont redondantes dans le modèle original MCM . L'ensemble réalisable de PL(λ) étant compact, $L_u(\lambda)$ est donc bien défini.

Remarquons que PL(0) correspond à la méthode de pénalité classique.

Sous des hypothèses aisément satisfaites ici, la valeur L^* est égale à la valeur optimale de MCM.

La fonction $L_u(x,\lambda)$ est une fois continûment différentiable en x. La deuxième différentielle présente des points de discontinuité.

2.1.6.2 L'algorithme

Bertsekas propose l'algorithme suivant pour résoudre PL(λ).

ETAPE 0 - Initialisation

Poser $\ell = 0$

Choisir $\lambda_\ell = 0$, $c_\ell > 0$.

ETAPE 1 - Résolution du sous-problème

Calculer x_ℓ solution optimale de PL(λ_ℓ).

ETAPE 2 - Ajustement des multiplicateurs

Poser $\lambda_{\ell+1} = \max \{0, \lambda_\ell + u_\ell (h - \Sigma\ x_a^k)\}$

poser $\ell = \ell + 1$ $\quad u_\ell \geq u_{\ell-1}$

aller à l'étape 1.

FIN

2.1.6.3 Propriété de convergence

Ironiquement, notre fonction objectif linéaire rend difficile l'analyse des propriétés de convergence de l'algorithme. En effet, si la fonction $L_u(x, \lambda)$ est strictement convexe en x, on peut travailler dans une théorie très riche. Cependant, dans le cas qui nous occupe, la fonction ne présente pas cette propriété, à cause de la linéarité de l'objectif initial.

Selon Bertsekas, [2 , chapitre 5], sous des conditions largement satisfaites ici la suite de points (x_ℓ, λ_ℓ) généré par l'algorithme présenté converge vers (x_*, λ_*) où x_* solution optimale de P et λ_* solution optimale de PL.

Cependant, il n'existe pas d'algorithme fini pour résoudre exactement $PL(\lambda_\ell)$ à l'étape 1. On devra donc résoudre $PL(\lambda_\ell)$ de façon approchée. Bertsekas étudie les propriétés de convergence d'un algorithme où x_ℓ est solution approchée de $PL(\lambda_\ell)$. Cependant, la fonction économique linéaire ne satisfait pas une hypothèse qui équivaut à la convexité forte de la fonction objective.

Il faut donc, au moins asymptotiquement, obtenir d'excellentes solutions pour le sous-problème PL(λ) pour espérer une convergence rapide.

2.1.6.4 Le paramètre de pénalité

Le choix de la suite des paramètres de pénalité doit faire l'objet d'un compromis. En effet, un facteur de pénalité u très grand assure convergence duale rapide et un sous-problème mal conditionné et vice-versa pour u petit. Il faut choisir entre beaucoup d'itérations faciles ou peu d'itérations chères.

Ce choix est aussi lié à la précision avec laquelle les sous-problèmes sont résolus, car on doit choisir entre résoudre précisément un problème facile ou résoudre de façon grossière un problème difficile.

2.1.6.5 Résolution du problème à coûts convexes

Le sous-problème PL(λ) est un problème de multiflot d'équilibre. Puisqu'il faut résoudre un tel problème à chaque itération, le succès de l'approche dépendra donc de l'existence d'un bon algorithme pour résoudre le multiflot d'équilibre. Nous allons étudier plusieurs méthodes proposées par Nguyen [15].

2.1.6.6 Critère d'arrêt

On peut envisager d'arrêter l'algorithme lorsque

$$\| \lambda_{\ell+1} - \lambda_\ell \| = \| \max \{ - \lambda_\ell, u_\ell (h - \sum_{k \in K} x_\ell^k) \} \| < \varepsilon$$

car cela correspond à un ajustement très lent des multiplicateurs. De plus, on a alors

$$\| \max \{0, h - \sum_{k \in K} x_\ell^k \} \| \leq \varepsilon / u$$

et le vecteur x_ℓ est alors très près d'être réalisable.

On peut, lors d'une seconde phase, élever le facteur de pénalité pour approcher davantage de l'ensemble réalisable à partir du point courant.

2.1.6.7 Problèmes non-réalisables

Si le problème initial n'admet pas de solution, alors

$$\max_{x \geq 0} L_u(\lambda) = -$$

Dans ce cas, la valeur objective et un sous-ensemble des multiplicateurs divergeront. On peut imaginer plusieurs mécanismes détectant ce fait en s'appuyant sur le fait que l'ensemble réalisable $PL(\lambda)$ est compact. Par exemple le problème est non-réalisable si

$$L_u(\lambda) > \max \{ \sum_{k \in K} c^k x^k \mid Ax^k = b^k, \ 0 \leq x^k \leq M \}$$

2.1.7 Les méthodes de direction de descente pour le modèle multiflot d'équilibre

L'application de la méthode du lagrangien augmenté au problème de multiflot compatible nécessite un sous-algorithme pour résoudre le problème relaxé, dont la fonction économique est convexe. Van Der Straatten-Tilquin [24] propose une adaptation de l'algorithme des arcs non-conformes.

Pour notre part, nous nous appuierons sur l'expérience de Nguyen [15], pour la résolution du problème de multiflot d'équilibre. Nguyen, dans sa thèse de doctorat, propose en effet plusieurs adaptations de la méthode de direction de descente tirant parti de la structure du problème.

2.1.7.1 La méthode de direction de descente

Soit le modèle

PG
$$\min f(x)$$
$$Ax = b$$
$$x \geq 0$$

où

f est convexe et continûment différentiable

$X = \{x \mid Ax = b,\ x \geq 0\}$ est compact

d est une direction de descente pour f en x, ssi

$$\exists\ \tau_{max} > 0 \qquad x + \tau d \in X \quad \forall\ 0 \leq \tau \leq \tau_{max}$$

et

$$\nabla f(x)\ d < 0$$

Posons, $D(x) = \{d \mid d \text{ direction de descente}\}$.

La méthode de direction de descente génère une suite de points

$$\{ x^{\ell} \}_{\ell=0}^{\infty} \subset X$$

selon la règle suivante.

ETAPE 0 - Initialisation

Choisir $x^0 \in X$

poser $\ell = 0$

ETAPE 1 - Construction de la direction de descente

Si $D(x) = \emptyset$, on a terminé

sinon, choisir $d^{\ell} \in D(x)$.

calculer $\tau_{max} = \max \{ \tau \mid x^{\ell} + \tau d^{\ell} \in X \}$

ETAPE 2 Recherche linéaire

Trouver τ^{ℓ} solution optimale de

PRU

$$\min_{0 \leq \tau \leq \tau_{max}} f(x^{\ell} + \tau d^{\ell})$$

ETAPE 3 - Construction du point suivant

Poser

$$x^{\ell+1} \leftarrow x^{\ell} + \tau^{\ell} d^{\ell}$$

$$\ell \leftarrow \ell + 1$$

Retourner à l'étape 1.

Remarquons que si $D(\overline{x}) = \emptyset$, alors $\forall\, x \in X$

$$f(\overline{x}) + \nabla f(\overline{x}) \quad (x - \overline{x}) \geq f(x)$$

D'où $\overline{x}$ optimal pour PG.

Nguyen démontre que la procédure précédente converge vers une solution optimale, si la règle qui détermine la direction de descente est une application fermée, i.e.

$$P: \; X \to Y \quad \text{est fermée ssi}$$

$$\left.\begin{array}{ll} x^k \to x & x^k \in X \\ y^k \to y & y^k \in P(x^k) \end{array}\right\} \Rightarrow y \in P(x)$$

Dans les pages suivantes, nous allons traiter du modèle multiflot suivant, légère généralisation du multiflot d'équilibre

$$\text{MEC} \qquad \min \sum_{a \in A} \left[\sum_{k \in K} c_a^k x_a^k + g_a \left(\sum_{k \in K} x_a^k \right) \right]$$

$$\left.\begin{array}{l} Ax^k = b^k \\ 0 \leq x^k \leq M \end{array}\right\} \qquad \forall\, k \in K$$

où $\quad c^k \geq 0 \qquad \forall\, k \in K$

où g_a continûment différentielle, strictement convexe, positive croissante, $\forall\, a \in A$.

La fonction objectif garantit que les bornes supérieures sur chaque flot seront inactives, si M est choisi suffisamment grand. Ces bornes nous permettent de raisonner sur un ensemble réalisable compact.

2.1.7.2 Méthode de Frank-Wolfe

A) Approximation linéaire

Dans le contexte du problème PG, soit $\bar{x} \in X$, considérons le modèle

PAL $\quad \min \quad \nabla f(\bar{x})\,(y - \bar{x}) + f(\bar{x})$

$\qquad\qquad y \in X$

Soit $\bar{y}$, solution optimale extrêmale de PAL. On sait que puisque X polytope borné, on peut toujours choisir une telle solution.

B) Direction réalisable

$\bar{d} = (\bar{y} - \bar{x})$ est une direction réalisable, car X est convexe et dès lors

$\forall\, \theta \in [0, 1] \quad \bar{x} + \theta\, \bar{d} = \bar{x} + \theta\,(\bar{y} - \bar{x}) = (1 - \theta)\,\bar{x} + \theta\,\bar{y} \in X$

C) Direction de descente

Soit x^* solution optimale de PG, $\bar{x} \in X$ et $\bar{y}$ solution de PAL associé à $\bar{x}$ par convexité de f et définition de $\bar{y}$

$$f(x^*) \geq f(\bar{x}) + \nabla f(\bar{x}) \quad (x^* - \bar{x})$$

et

$$\nabla f(\bar{x}) \ (x^* - \bar{x}) \geq \nabla f(\bar{x}) \ (\bar{y} - \bar{x})$$

D'où

$$f(\bar{x}) \geq f(x^*) \geq f(\bar{x}) + \nabla f(\bar{x}) \ (\bar{y} - \bar{x})$$

Il est clair que, ou bien $\bar{x}$ est une solution optimale, ou bien $\bar{d}$ est une direction de descente.

La résolution de PAL nous fournit donc un encadrement de la valeur optimale. Cela nous fournit un critère d'arrêt naturel pour l'algorithme à l'étape 1. On cesse d'itérer lorsque pour ε fixé,

$$f(\bar{x}) - f(x^*) \leq -\nabla f(\bar{x}) \ (\bar{y} - \bar{x}) < \varepsilon$$

D) Application au multiflot d'équilibre, la méthode globale.

La méthode globale est l'application de la méthode de Frank et Wolfe au problème de multiflot ME. Le problème de recherche de direction devient

$$\text{MPAL} \qquad \min \sum_{k \in K} \sum_{a \in A} [c_a^k + \nabla g_a \left(\sum_{k \in K} x_a^k \right)] \ y_a^k$$

$$\text{s.a.} \qquad A \, y^k = b^k \qquad k \in K$$

$$0 \leq y^k \leq M$$

Sous l'hypothèse qu'il n'existe pas de circuits à coûts négatifs, les bornes supérieures ne sont pas actives et ce problème est séparable en K problèmes de plus court chemin. Sinon, il s'agit de problème de flot à coût minimum.

Le problème de recherche unidimensionnelle devient

MPRU

$$\min_{0\leq\theta\leq 1} \quad (1-\theta)\, c_x + c_y + \sum_{a\in A} g_a((1-\theta)\,\tilde{x}_a + \theta\,\tilde{y}_a)$$

où

$$c_x = \sum_{k\in K} c^k x^k$$

$$c_y = \sum_{k\in K} c^k y^k$$

$$\tilde{x}_a = \sum_{k\in K} x_a^k$$

$$\tilde{y} = \sum_{k\in K} y_a^k$$

Remarquons que seul les flots totaux sont impliqués dans la recherche de direction et la recherche linéaire. Cet algorithme a l'avantage de ne nécessiter que la connaissance des flots totaux sur chaque arc pour fonctionner.

L'algorithme résultant s'énonce:

ETAPE 0 - Initialisation

Choisir $(x^k)_{k\in K}$ réalisable pour MEC.

ETAPE 1 - Approximation linéaire

Calculer $(y^k)_{k\in K}$ solution optimale de MPAL

si $\sum_{k\in K} \sum_{a\in A} [c_a^k + \nabla g_a (\sum_{k\in K} x_a^k)] [y_a^k - x_a^k] < \varepsilon$, terminer.

ETAPE 2 - Recherche linéaire

Trouver θ^* minimum de MPRU

ETAPE 3 - Calcul du nouveau point

Calculer

$$x \leftarrow (1 - \theta^*)\ x + \theta^*\ y$$

retourner à l'étape 1.

FIN

E) Application au multiflot, la méthode séquentielle.

Considérons le modèle obtenu en ne considérant comme variable que le flot d'une commodité $\bar{k}$ fixée, nous obtenons le modèle

$$\text{MEC}(\bar{k}) \qquad \min \quad \sum_{a\in A} \sum_{k\in K} c_a^k x_a^k + \sum_{k\in K} g_a (\sum_{k\in K} x_a^k)$$

$$A\, x^{\bar{k}} = b^{\bar{k}}$$

$$0 \le x^{\bar{k}} \le M$$

$$x^k \quad \text{fixé si} \quad k \neq \bar{k}$$

On peut appliquer la méthode de Frank et Wolfe à ce modèle. Cet algorithme se déduit de façon parallèle à l'algorithme global.

Si l'on suppose connue $(x^k)_{k\in K}$ une solution réalisable de MEC. On obtient l'algorithme suivant.

ETAPE 1 - Approximation linéaire

Calculer $y^{\overline{k}}$ solution optimale de

$$\min \sum_{a\in A} [c_a^k + \Delta g_a(\sum_{k\in K} x_a^k)]\, y_a$$

$$Ay = b^{\overline{k}}$$

$$0 \le y \le M$$

si $\sum_{a\in A} [c_a^k + \nabla g_a (\sum_{k\in K} x_a^k)] + [y_a^{\overline{k}} - x_a^{\overline{k}}] < \varepsilon/K$

on a terminé.

ETAPE 2 - Recherche linéaire

Calculer θ^* solution optimale de

$$\min_{0\le\theta\le 1} c^{\overline{k}}((1-\theta)\, x^{\overline{k}} + \theta\, y^{\overline{k}}) + \sum_{a\in A} g(\sum_{k\in K} x_a^k + \theta\,(y^{\overline{k}} - x^{\overline{k}}))$$

ETAPE 3 - Nouveau point

Poser

$$x^{\overline{k}} \leftarrow (1-\theta^*)\, x^{\overline{k}} + \theta^*\, y^{\overline{k}}$$

Retour à l'étape 1.

FIN

A partir de l'algorithme précédent, nous pouvons construire l'algorithme séquentiel qui s'énonce comme suit.

ETAPE 0 - Initialisation

Choisir $(x^k)_{k\in K}$ solution réalisable de MEC

poser $L = K$.

ETAPE 1 - Critère d'arrêt

Si $L \neq 0$, on a terminé

sinon, on choisit $\overline{k} \in L$.

ETAPE 2 - Résolution de MFCP

Appliquer l'algorithme énoncé précédemment pour la commodité $\overline{k}$.

ETAPE 3 - Revision de la liste

Si le flot $x^{\overline{k}}$ n'a pas été modifié

alors $L = L - \{\overline{k}\}$

$L = K$

on retourne à l'étape 1.

FIN

On justifie cet algorithme en remarquant que la direction $d^{\bar{k}}$ définie à l'étape 1 du sous-algorithme correspond à une direction de descente d, pour le problème global MEC

$$d^k = \begin{cases} y^{\bar{k}} - x^{\bar{k}} & \text{si} \quad k = \bar{k} \\ 0 & \text{si} \quad k \neq \bar{k} \end{cases}$$

De plus, si le sous-algorithme ne génère pas de direction de descente la commodité $\bar{k}$ est retirée de la liste L. Donc, on ne peut passer qu'un nombre fini de fois à l'étape 2 sans générer une direction de descente ou stopper à l'étape 1. Remarquons aussi qu'à l'arrêt, l'ensemble L étant vide:

$$\sum_{k \in K} \sum_{a \in A} [c_a^k + \nabla g_a(\Sigma x_a^k)] [y_a^k - x_a^k] < k \cdot \varepsilon / K = \varepsilon$$

Ceci correspond au critère d'arrêt de la méthode globale.

2.1.7.3 Méthode du gradient réduit

A) Partition des variables

Supposons que A, la matrice des contraintes de PG soit de plein rang.

Permutant les colonnes de A, on peut sans perte de généralité, partitionner A et le vecteur des variables x selon...

$$A = (B \mid N) \qquad x = (x_B, x_N)$$

On peut alors écrire un nouveau modèle qui ne dépend que des variables x_N, équivalent à PG. Posons

$$g(x_N) = f(B^{-1}[b - Nx_N] \quad , x_N]$$

On obtient le modèle

PGB $\qquad \min \quad g(x_N)$

$$B^{-1}[b - Nx_N] \geq 0$$

$$x_N \geq 0$$

B) Direction de descente

Nous allons supposer, dans le développement qui suit, que l'hypothèse de non-dégénérescence est vérifiée, c'est-à-dire

$$B^{-1}[b - N x_N] > 0$$

La construction d'une direction de descente en x_N est alors très facile, car on peut construire pour $\varepsilon > 0$ une boule ouverte centrée en x_N $O(x_N, \varepsilon)$ telle que

$$O(x_N, \varepsilon) \subset \{ x_N \mid B^{-1}[b - Nx_N] > 0 \}$$

On peut considérer que localement les contraintes de positivité des variables de base sont inactives. Dès lors, toute direction d_N satisfaisant

$$d_{N_i} \geq 0 \quad \text{si} \quad x_{N_i} = 0$$

est réalisable, car on peut trouver τ_{max} suffisamment petit pour que

$$B^{-1} [b + N(x_N + \tau_{max}\, d)] \geq 0$$

Une variante de la méthode du gradient réduit propose la direction de descente suivante d^{GR}

$$d^{GR}_{N_i} = \begin{cases} - \dfrac{\nabla g(x_N)}{\nabla x_{N_i}} & \text{si} \quad \dfrac{\nabla g(x_N)}{\nabla x_{N_i}} < 0 \\ - \dfrac{\nabla g(x_N)}{\nabla x_{N_i}} \cdot x_{N_i} & \text{si} \quad \dfrac{\nabla g(x_N)}{\nabla x_{N_i}} \geq 0 \end{cases}$$

La méthode du simplexe convexe, d'autre part, calcule la direction de descente en choisissant d'augmenter ou de diminuer la valeur d'une variable hors-base, selon un critère similaire au critère d'entrée du simplexe.

Posons $$\bar{c}_{N_i} = \frac{\nabla g(x_N)}{\nabla x_{N_i}}$$

$$r_{N_{i+}} = \min_{i \in N} \{ \bar{c}_{N_i} \}$$

$$r_{N_{i-}} = \max_{\in} \{ \bar{c}_{N_i}\, x_{N_i} \}$$

$$\left| r_{N_{i*}} \right| = \max \{ -r_{N_{i+}} , r_{N_{i-}} \}$$

La direction de descente d^{SC} est alors définie par

$$d^{SC} = \begin{cases} e_{i^*} & \text{si} \quad i^* = i^+ \\ -e_{i^*} & \text{si} \quad i^* = i^- \end{cases}$$

On peut interpréter ce choix de direction comme une méthode de "coordinate descent" pour les variables hors-bases.

Par construction, les deux directions sont des directions de descente sous l'hypothèse de non-dégénérescence. De plus, raisonnant sur l'ouvert $O(x_N, \varepsilon)$ si $d^{SC} = 0$ ou $d^{GR} = 0$. on a

$$\bar{c}_{N_i} = 0 \qquad \text{si} \quad x_{N_i} > 0$$

$$\bar{c}_{N_i} \geq 0 \qquad \text{si} \quad x_{N_i} = 0$$

D'où x_N optimal pour PGB et $(B^{-1}[b - N_{x_N}], x_N)$ optimal pour PG.

Nguyen démontre que sous l'hypothèse de non-dégénérescence, le processus de génération d'une telle direction est une application fermée.

C) L'algorithme

Bien sûr, l'hypothèse de non-dégénérescence ne tient pas en général et nous devons composer avec la possibilité que

$$B^{-1}[b - N(x_N + \tau d)] \not\geq 0 \qquad \forall \tau > 0$$

c'est-à-dire que

$$\exists\ i \in B\ ,\ \text{tel que}\ \ [B^{-1}\,[b - Nx_N\,]] = 0$$

Cela implique la possibilité d'un pas maximal nul. Pour sortir de ce cul-de-sac, il faut modifier la partition des variables en choisissant une nouvelle base. On obtient alors l'algorithme.

ETAPE 0 - Initialisation

Choisir $x \in X$

choisir B, une base de A.

ETAPE 1 - Direction de descente

Calculer d^{GR} ou d^{SC}, une direction de descente

si $d = 0$, on a terminé.

ETAPE 2 - Calcul du pas maximal

Calculer

$$\tau_{max} = \max \{\, \tau \mid B^{-1}\,[\, b - N\,(x_N + \tau\ d_N)] \geq 0\,\}$$

si $\tau_{max} = 0$, aller à l'étape 5.

ETAPE 3 - Recherche linéaire

Calculer

τ^* solution optimale de

$$\min_{0 \leq \tau \leq \tau_{max}} g(x_N + \tau\, d_N)$$

ETAPE 4 - Nouvelle solution

$$x_N \leftarrow x_N + \tau^* d_N$$

$$x_B \leftarrow B^{-1} [b - Nx_N]$$

retour à l'étape 1.

ETAPE 5 - Nouvelle base

Choisir i tel que $[B^{-1} [b - N(x_N + \tau d_N)]i < 0 \ \forall \tau > 0$.

Effectuer une itération duale du simplexe pour sortir i de la base. B devient alors, pour j bien choisi

$$B \leftarrow B - \{i\} \cup \{j\}$$

retour à l'étape 1.

Cet algorithme converge si on peut prouver que l'on ne peut faire qu'un nombre fini de pivots entre la construction de deux directions réalisables et la modification de la solution courante.

D) Le simplexe convexe appliqué au problème de multiflot

Considérons le modèle MEC($\overline{k}$) introduit en 2.1.7.3 où un seul flot $\overline{k}$ est considéré comme variable, les autres étant tenus fixés. L'application du simplexe convexe à ce modèle donne lieu à de nombreuses simplifications, à cause de la structure de réseau et du fait que les coûts sont séparables par arc. Négligeons l'indice $\overline{k}$ pour alléger la notation.

Une base B de A correspond ici à un arbre sur le réseau. Chaque arc hors-base forme un cycle fondamental avec les arcs de la base. Avec une structure de donnée appropriée, il est possible de retracer les arcs du cycle rapidement.

Nous noterons δ^a le vecteur caractéristique associé à un arc a hors-base, c'est-à-dire

$$\delta_i^a = \begin{cases} 1 & \text{si i est orienté dans le sens de a sur le cycle} \\ -1 & \text{si i est en sens contraire} \\ 0 & \text{si i n'est pas du cycle} \end{cases}$$

Nous allons décrire étape par étape la transposition de l'algorithme. Commençons par l'évaluation de la différentielle à l'étape 1 qui devient ici:

$$\bar{c}_{N_a} = \frac{\nabla g\ (x_N)}{\nabla\ x_{N_a}} = \sum_{i \in A} [\ c_i + \nabla\ g_i\ (\sum_{k \in K} x_i^k)\ ]\ \ \delta_i^a$$

On peut calculer ces quantités en retraçant les cycles sur le réseau, mais il est beaucoup plus économique de calculer

$$\bar{c}_{N_a} = [\ c_a + \nabla\ g_a\ (\sum_{k \in K} x_a)\ ] + \pi_{i(a)} - \pi_{j(a)}$$

où les quantités u_i sont solutions du système

$$\bar{c}_{N_a} = [c_a + \Delta g_a(\sum_{k\in K} x_a^k)] + \pi_{i(a)} - \pi_{j(a)} = 0, \ \forall\, a\in B \qquad (2.11)$$

$$\pi_s = 0 \qquad (2.12)$$

Ces quantités correspondent aux variables duales du simplexe sur le réseau et se calculent en parcourant l'arbre de base selon une fonction de traverse en pré-ordre.

On a donc une méthode efficace pour calculer d^{SC} à l'étape 1. Considérons maintenant le problème du pas maximal à l'étape 2.

Le fait de modifier le flot x_a se répercutera sur la base en modifiant le flot sur le cycle fondamental, on peut définir l'intervalle compact pour lequel le flot est réalisable par la formule

$$[\ \alpha_{min}\ ,\ \alpha_{max}\ ] = \{\ \alpha \mid\ 0 \le x\ + \delta^a\ \alpha \le M\ \}\ .$$

Ces quantités se calculent en retraçant le cycle fondamental associé à a et l'on pose ensuite

$$\tau_{max} = \begin{cases} \alpha_{max} & \text{si} \quad \bar{c}_{N_a} < 0 \\ -\alpha_{min} & \text{si} \quad \bar{c}_{N_a} > 0 \end{cases}$$

Passons à l'étape 3, où le sous-problème à résoudre s'écrit

$$\min f(\alpha) = \sum_{i \in A} [c_i^{\overline{k}} [x_i + \alpha \delta_i^a] + g_i(\sum_{k \in K} x_i^k + \alpha \delta_i^a)$$

$$\text{s.a.} \qquad \alpha_{min} \leq \alpha \leq \alpha_{max} .$$

La différentielle de la fonction économique s'écrit pour α fixé

$$\frac{\nabla f(x)}{\nabla \alpha} = \sum_{i \in A} c_i \delta_i^a \alpha + \sum_{i \in A} \delta_i^a \cdot \nabla g_i(\sum_{k \in K} x_i^k + \alpha \delta_i^a).$$

Dès lors, on voit que le sous-problème se résout en ne considérant que le cycle fondamental et la différentielle est évaluée en retraçant et en sommant les composantes.

La modification de flot à l'étape 4 se fait aussi en retraçant le cyle et s'écrit

$$x \leftarrow x + \alpha^* \delta^a$$

Pour terminer la description de l'algorithme, il reste à décrire le processus de changement de base à l'étape 5. Nous allons utiliser la structure de la matrice de demande pour définir une procédure simple qui nous assure un nombre fini de pivots successifs. Cette méthode est due à Nguyen [15].

E) Règle de pivotage

Supposons que la base B est une arborescence de racine s. Notons que si une telle base n'existe pas, le problème est irréalisable. La règle de pivotage est la suivante.

Soit p, l'arc choisi à l'étape 1. Puisque B est une arborescence, le cycle fondamental formé par p avec B est constitué de deux chemins distincts, disons L_1 et L_2 partant d'une racine commune t et menant à j(p). Choisissons $p \in L_1$. La figure 3.5 est un exemple d'une telle construction.

Notons $\{a_i\}_{i=1}^{I^2}$ [$\{b_i\}_{i=1}^{I^1}$] la liste ordonnée des arcs de L_2[L_1] décrivant le chemin de j(p) à t. Nous avons alors

$$j(a_{i+1}) = i(a_i) \qquad \forall\, i \;\; 1 \le i < I^2$$

$$j(b_{i+1}) = i(b_i) \qquad \forall\, i \;\; 1 \le i < I^1 .$$

Supposons que $\tau_{max} = 0$, alors la règle distingue deux cas

i) si $\bar{c}_p < 0$, alors

$$\tau_{max} = \max \{ \tau \mid x + \tau \delta^p \}$$

$$= \min \{ x_{a_i} \mid 1 \le i \le I^2 \}$$

soit ℓ tel que

$$\ell = \min \{ i \mid x_{a_i} = 0 , \;\; 1 \le i \le I^2 \}$$

nous distinguons de nouveau deux cas

i.a) si $\ell = 1$ alors, dans ce cas, nous avons $j(a_1) = j(p)$ et nous effectuons le pivot

$$B \leftarrow B - \{a_1\} + \{p\}$$

i.b) si $\ell \neq 1$, il faut calculer

$$Q = \{q \mid x_q > 0 \quad \text{et } j(q) = j(a_\ell)\}$$

et pour $q \in Q$, on effectue le pivot

$$B \leftarrow B - \{a_\ell\} + \{q\}$$

ii) si $\bar{c}_j > 0$, alors

$$\tau_{max} = \max \{\tau \mid x - \tau \delta^p \geq 0\}$$
$$= \min \{x_{b_i} \mid 1 \leq i \leq I^1\}$$

si 1 est choisi comme en i, soit

$$\ell = \min \{i \mid x_{b_i} = 0, \ 1 \leq i \leq I^1\}.$$

Nous avons $\ell > 1$, car $x_{b_1} = x_p > 0$ et nous pouvons choisir q comme en i.b pour effectuer le pivot

$$B \leftarrow B - \{a_\ell\} + \{q\}.$$

L'arc a_ℓ est bien défini par l'hypothèse. C'est le premier arc du chemin L_1 [L_2] de flot insuffisant rencontré lorsque l'on remonte le chemin L_1 [L_2] de $j(p)$ vers t.

Pour que la règle soit bien définie, il faut montrer que Q n'est jamais vide. L'existence de $q \in Q$ est garantie par le choix de ℓ la conservation de flot et la nature de la matrice de demande. On peut montrer que dans le cas i.b (on peut faire le même raisonnement dans le cas (ii).

$$\sum_{\substack{j(a) = j(a_\ell) \\ a \neq a_\ell}} x_a \geq b_{j(a_\ell)} \quad x_{a_{\ell-1}} > 0$$

De plus, il est clair que la nouvelle base B^* est une arborescence. Le dernier point à considérer est l'élimination du cyclage. Introduisons les fonctions

$$U : P(A) \to \mathbb{R} \quad , \quad T : P(A) \to \mathbb{N}$$

$$T(B) = | \{ a \in B \mid x_a = 0 \} |$$

$$U(B) = \sum_{i \in N} u_i$$

où u_i est solution du système d'équations (2.11-12). T correspond à la quantité de dégénérescence et U a la somme des potentiels sur les noeuds qui correspondent eux-mêmes aux longueurs des chemins dans l'arborescence.

Dans le cas i.a, nous avons

$$U(B^*) < U(B)$$

$$T(B^*) \leq T(B)$$

Et dans le cas i.b et ii

$$T(B^*) < T(B) .$$

Les arborescences distinctes étant en nombre fini et les inégalités précédentes empêchant le retour d'une arborescence, un nombre fini de pivots produira une direction de descente.

F) Critère d'arrêt

Bien sur, les sous-problèmes ne sont pas résolus à l'optimalité et on cesse d'itérer lorsque pour ε_0 fixe

$$\| d^{SC} \|^2 = | r_{N_{i^*}} | < \varepsilon_0 .$$

De plus, on conviendra qu'un arc a a un flot nul lorsque pour τ_0 fixe

$$x_a < \tau_0 .$$

Ce critère intervient lors du choix de la direction de descente et lors de la détermination de pas maximal.

G) L'algorithme séquentiel

Nous avons montré comment la méthode du simplexe convexe pouvait être appliquée à la résolution (approximative compte tenu des paramètres ε_0 et τ_0). du modèle MEC($\overline{k}$). L'algorithme AS présenté en 2.1.7.3 peut être utilisé pour résoudre le modèle MEC en remplaçant la méthode de Frank-Wolfe par la méthode du simplexe convexe à l'étape 2.

Chaque direction de descente générée peut être considéré comme une direction de descente pour le modèle initial MEC. La méthode s'apparente donc à une technique de résolution par sous-ensemble ("Multiple pricing") couramment utilisée en programmation linéaire. Chaque flot correspond ici à un sous-ensemble.

2.2 Analyse comparative

Nous avons présenté, dans la première partie de ce chapitre, plusieurs méthodes de solution pour le problème qui nous intéresse. Nous allons poursuivre notre recherche d'un algorithme efficace par une analyse critique des méthodes discutées précédemment. Nous comparerons d'abord les propriétés théoriques des différentes approches, puis nous résumerons l'expérience numérique connue.

2.2.1 Propriétés théoriques

Nous discuterons d'abord des méthodes exactes qui utilisent toutes le simplexe d'une façon ou d'une autre.

Considérons d'abord la méthode de décomposition de Dantzig-Wolfe (que nous dénoterons DW, pour alléger le discours). L'intérêt de DW est de remplacer le modèle original par un modèle ayant beaucoup moins de contraintes. Nous passons en effet de $nK+m$ à $K+m$ contraintes. Relaxant les con-

traintes de capacité inactives et utilisant la méthode de "généralized upper bounding" pour gérer les contraintes de convexités, la taille de la matrice à inverser est égale au nombre de contraintes de capacité actives près de l'optimum, disons p ($p < m$).

Le problème de génération de colonnes est ici d'implantation facile, des algorithmes efficaces existants pour le problème de plus court chemin.

De son côté, la méthode de "Generalized upper bounding" (GUB) n'est finalement qu'une façon de résoudre les systèmes d'équations linéaires nécessaires au fonctionnement du simplexe dans le but de limiter la taille de la matrice à inverser. De ce point de vue, GUB est efficace ramenant à p la taille de l'inverse explicite.

Cependant, chaque itération implique le calcul de plusieurs produits scalaires et surtout, nécessite la connaissance d'une base sur le réseau pour chaque flot. Cela implique $O(Kn)$ informations.

Le problème relaxé PR résolu à l'étape 1 de l'algorithme de partition primale de Rosen (ROSEN) n'a que m (possiblement p) contraintes et $K(m-n)$ variables. Il faut donc savoir combien de fois il devra être résolu.

Grigoriadis et White [6] proposent d'exploiter toute information duale disponible. En effet, si nous calculons une approximation des multiplicateurs optimaux λ^* associés aux contraintes de capacité, nous pouvons faci-

lement déterminer de bonnes bases de départ pour chaque flot. Les colonnes correspondantes à ces bases dans le programme-maître de DW aurait de bonnes chances d'être de base à l'optimum pour DW. Ce sont donc de bonnes bases, car on peut penser que tous leurs arcs seront à flot non négatifs à l'optimum de PR.

Une telle approximation des λ^* peut être obtenue comme proposée en 2.1.4 en appliquant la méthode de Bender au dual de MCM. L'algorithme de sous-gradient décrit en 2.1.4.4 (dénoté BSG) peut fournir une approximation à peu de frais.

Nous irons plus loin en proposant d'utiliser une solution approximative en (x, λ) obtenue par la méthode du lagrangien augmenté (LA) pour initialiser ROSEN à l'étape 0. Si le simplexe convexe (SC) est utilisé pour résoudre les sous-problèmes de flot à coûts convexes, nous obtenons de bonnes bases de départ et les arcs hors-base susceptibles de porter un flot non-nul à l'optimum.

Sur un autre plan, Grigoriadis et White proposent d'utiliser la similitude entre ROSEN et GUB pour limiter le coût de réinitialiser PR. En effet, ROSEN peut être vu comme une approche primale-duale pour résoudre le modèle original. Il est possible, lors d'un changement de base à l'étape 3, de calculer l'inverse de la base courante du nouveau problème PR en appliquant la règle de pivotage 2a) de GUB (voir 2.1.5.4).

Dans ce cas, l'inverse de la base courante est mis-à-jour en le prémultipliant par une matrice élémentaire-colonne pour chaque pivot exécuté à l'étape 3 de ROSEN.

Si PR est résolu à l'aide d'une technique de factorisation de la base, le passage à l'étape 3 se résume à ajouter des facteurs à la base courante. L'algorithme résultant semble très prometteur.

La dernière méthode exacte que nous entendons discuter est la méthode de linéarisation externe pour résoudre le modèle d'allocation de ressources (ARLE) présenté en 2.1.3.3. Notons d'abord que le modèle de linéarisation externe a autant de variables que MCM. De plus, on ne peut prévoir le nombre de contraintes générées et surtout combien d'entre elles seront actives à l'optimum.

La nature combinatoire du simplexe devient un inconvénient lorsque la taille des problèmes traités augmente. Dans le pire cas, nous pourrions épuiser toutes les bases de la matrice de contraintes. Les observations empiriques indiquent que le nombre moyen d'itérations nécessaires est généralement égal à quelques fois le nombre de contraintes et cela implique déjà beaucoup de travail.

Selon ce critère, GUB est largement défavorisée et comme nous le verrons, n'a pas fait l'objet d'une implantation.

DW et ROSEN permettent toutes deux de travailler sur un seul flot à la fois et donc de traiter de très grands problèmes. DW a l'avantage d'être de nature primal sur ROSEN qui se termine en vérifiant la réalisabilité. Cependant, de nombreux auteurs ont signalé que DW converge très lentement et fournit de mauvaises bornes inférieures.

Nous avons déjà signalé comment l'information générée par BSG pouvait être utilisée pour accélérer ROSEN. BSG fournit aussi une borne inférieure pour l'algorithme de sous-gradient affecté au modèle d'allocation de ressources (ARSG). La méthode de sous-gradient a des propriétés de convergence faible et la vitesse de convergence peut être influencée par la qualité de la borne inférieure. (Remarquons que la convergence de BSG dépend à son tour de la connaissance d'une bonne borne supérieure).

Remarquons que le calcul du sous-gradient est un problème séparable par flot et que le sous-problème de projection est séparable par arcs. La matrice d'allocation est donc accédée à chaque itération, d'abord par flot, puis par arc. Cela force la présence en mémoire de la matrice soit Km informations. Cet algorithme est donc très exigeant sur le plan mémoire.

Le succès de la méthode du lagrangien augmenté repose quand à lui sur l'efficacité des algorithmes de résolution du problème de multiflot d'équilibre. Nous avons décrit en 2.1.7 plusieurs algorithmes qui ont connus un certain succès lorsqu'appliqués à des problèmes réels de grande taille. Il

s'agit cependant de méthodes du premier ordre très sensibles au conditionnement de la fonction économique. Dans notre cas, nous pouvons choisir la forme de la fonction économique en fixant le paramètre de pénalité. Il reste à démontrer qu'il existe un intervalle pour le paramètre de pénalité qui assure à la fois convergence en x et en λ.

LA décompose naturellement le problème flot par flot. Les problèmes de multiflot d'équilibre sont résolus flot par flot. Cela permet l'usage efficace de la mémoire secondaire pour traiter de grands problèmes.

AR a cependant l'avantage d'être de nature primale sur LA où la solution en x demeure irréalisable tout au long de l'algorithme.

2.2.2 Résultats numériques

Nous allons discuter quatre sources de résultats numériques disponibles dans la littérature. Il s'agit des thèses de doctorat de Swoveland [22] et Van der Straatten-Tilquin [24] et de deux articles dus, d'une part à Grigoriadis et White [6], d'autre part à Kennington et Shalaby [11].

2.2.2.1 La thèse de Swoveland

Swoveland s'intéresse aux problèmes de production-distribution et de communication. L'auteur considère des bornes inférieures et supérieures pour le flot sur chaque arc. Il considère aussi des taux de consommation

de la capacité différent pour chaque arc et chaque flot. Les graphes sont très peu denses. Le modèle s'écrit

$$\min \sum_{k \in K} c^k x^k$$

$$A X^k = b^k \qquad \forall\, k \in K$$

$$x^k \geq 0 \qquad \forall\, k \in K$$

$$\ell \leq \sum_{k \in K} q^k I x^k \leq h$$

où $0 \leq \ell \leq h$ et $q^k \geq 0, \quad \forall\ k \in K$

L'auteur étudie deux méthodes exactes, soit ARLE et DW. Dans le cas de ARLE, les sous-problèmes de flot à coût minimum sont résolus par l'algorithme des arcs non-conformes (out-of-kilter). DW est appliquée selon deux stratégies. Selon la première stratégie, chaque flot correspond à une paire origine-destination. Selon la seconde, chaque flot correspond à une origine.

L'auteur, pour résoudre le problème de génération de colonne, utilise un algorithme de programmation dynamique adapté aux graphes avec peu de cycles. Cette routine est codée en langage-machine et est dix fois plus rapide que la version écrite en FORTRAN. Cette routine utilisant 20% des temps de calcul, son équivalent FORTRAN utiliserait 70% d'un temps total 2.8 fois plus élevé. Cela met en danger l'interprétation faite des résultats.

Pour alléger le discours, nous dénoterons la taille d'un problème par le quadruplet (n, p/m, k) ou le triplet (n, m, k).

Les résultats numériques concernent six problèmes de production-distribution et trois problèmes de communication. Les tailles vont de (6,4/9, 2) à(356, 177/1279, 6) pour les problèmes de production-distribution et de (5, 3/12, 5) à (15, 18/48, 15) pour les problèmes de communication.

L'implantation de ARLE n'a pu résoudre efficacement un problème de taille (37, 9/67, 3). Elle génère trop de contraintes et le simplexe consomme trop de temps. Cette approche s'est avérée inefficace.

Les performances de DW sont comparativement excellentes. Sur les problèmes de communication, la décomposition par origine est très performante. La convergence est lente, mais les bornes produites sont excellentes. Sur les problèmes de production-distribution, la décomposition par paire origine-destination converge rapidement, mais la qualité des bornes inférieures est mauvaise.

Les tests ont été réalisés sur IBM 360/67. Les langages utilisés étaient FORTRAN et ASSEMBLER 360. L'auteur disposait d'une mémoire de 1M mots. Le plus grand problème a été résolu en 608 secondes.

2.2.2.2 La thèse de Van der Straatten-Tilquin

L'auteur considère le modèle de multiflot compatible muni d'une fonction économique quadratique. Pour Q_{mxm} positive définie, c'est le modèle:

$$\min \quad y^T Q y + cy$$

$$\text{s.a.} \quad \sum_{k \in K} x^k = y$$

$$Ax^k = b^k \qquad \forall k \in K$$

$$x^k \geq 0 \qquad \forall k \in K$$

$$\sum_{k \in K} x^k \leq h$$

L'auteur a codé trois algorithmes pour fin de comparaison, soit ROSEN, ARSG et LA.

Les deux dernières méthodes nécessitent un sous-algorithme pour résoudre des problèmes de flot à coût convexe sous contraintes de capacité. L'auteur utilise une généralisation de l'algorithme des arcs non-conformes (dénoté CCU). Cet algorithme est bien adapté à la réoptimisation primale et duale. Largement testé, CCU s'est avéré efficace et robuste.

Le programme-maître PR de ROSEN se déduit de façon similaire au cas linéaire. Il est résolu par l'algorithme de Lemke,car sa fonction économique est quadratique.

L'implantation de ARSG est celle décrite en 2.1.3, sauf que les sous-problèmes de flot à coût quadratique sont résolus avec CCU. L'implantation de LA se distingue de l'énoncé de 2.1.6 du fait que les multiplicateurs λ sont ajustés très souvent, bien avant que le problème PL(λ) soit résolu. On obtient ainsi plus rapidement de bons multiplicateurs. Chaque arc a un facteur de pénalité différent déduit des coefficients de la matrice Q.

On trouve des résultats numériques pour des problèmes de taille (17, 26, 35) et (25, 44, 3) pour lesquels demandes, capacités et coûts sont générés aléatoirement. Les algorithmes sont codés en PASCAL et exécutés sur CYBER 173. Les temps de calcul atteignent typiquement quelques centaines de secondes.

ROSEN s'avère très coûteux utilisant beaucoup d'espace et de temps à cause de l'algorithme de Lemke. AR s'avère compétitif avec LA, mais moins fiable. AR nécessite une bonne connaissance préalable de la nature du problème pour bien performer.

2.2.2.3 L'article de Grigoriadis et White

L'article décrit l'application de ROSEN au modèle MCM. On y trouve aussi les raffinements que nous avons résumés en 2.2.1. Cependant, l'implantation utilisée pour les tests est relativement grossière. A chaque retour à l'étape 1, le problème PR est construit explicitement, réinitialisé et résolu à l'optimalité.

Le modèle considéré est une généralisation du problème de transport. Nous distinguons plusieurs sources et plusieurs puits sur lesquels on construit un graphe bipartite complet (un arc pour chaque paire source-puits). A chaque flot correspond un vecteur d'offres émanant des sources et de demandes sur les puits.

Les auteurs ont résolu huit problèmes de taille (8, 15, 10-20) et (21, 98, 5-10). L'algorithme a été codé en FORTRAN et les tests effectués sur IBM 360. Cette implantation s'est avérée 10 à 20 fois plus rapide qu'un simplexe standard.

2.2.2.4 L'article de Kennington et Shalaby

Les auteurs sont préoccupés par le même modèle que Grigoriadis et White (voir 2.2.2.3). Ils proposent ARSG comme méthode de solution. L'algorithme est lancé pour un nombre fixe d'itérations et l'on calcule le

temps nécessaire pour obtenir une certaine précision sur l'optimum connu. Les temps obtenus sont comparés avec les temps nécessaires pour résoudre à l'optimalité le problème à l'aide de APEX III, un code de simplexe commercial et d'une implantation de ROSEN dus aux mêmes auteurs. L'article ne comporte aucun détail sur ce dernier code.

La taille des problèmes traités va de (16, 64, 8) à (24, 144, 12) et sont générés aléatoirement. Le code, écrit en FORTRAN, est exécuté sur CYBER 72. Les temps de solution atteignent quelques centaines de secondes.

ARLE s'avère 50% plus efficace que ROSEN pour une précision de 1 à 5%. L'auteur se base sur ces résultats pour assurer que ARSG est la méthode la plus prometteuse pour de grands problèmes.

2.2.3 Conclusion

Il est difficile de tirer des conclusions de l'expérience numérique connue. Les données disponibles concernent en effet des problèmes de type divers, des implantations de qualité fort diverses. De plus, la majorité des problèmes-tests sont de petite taille. Prédire les performances des divers algorithmes en extrapolant à partir de problèmes de petite taille est un procédé pour le moins incertain.

Il nous apparaît cependant qu'aucune méthode exacte ne semble applicable pour résoudre des problèmes de grande taille. DW pourrait sans doute être utilisé comme heuristique, quoique les coûts de génération de colonnes risquent de devenir prohibitif à mesure que la taille du problème augmente. ROSEN pourrait vraisemblement terminer la résolution d'un problème qu'une méthode heuristique aurait sérieusement entamée comme indiqué en 2.2.1. Dans ce dernier cas, il faudrait investir beaucoup de travail dans la réalisation d'une implantation sophistiquée.

Il reste deux candidats sur les rangs pour la construction de solution approchée pour le modèle MCM, soit ARSG et LA. Ces méthodes représentent deux écoles de pensée plus ou moins opposées, soit respectivement la décomposition par les ressources et par les prix. Les résultats de Kennington-Shalaby [11] pour ARSG nous apparaissent peu encourageants. De plus, les tests effectués par Van der Straatten-Tilquin ne permettent pas d'établir la supériorité d'une approche sur l'autre.

LA se distingue de ARSG au niveau de ses besoins en mémoire centrale. Nous avons déjà montré en 2.2.1 que ARSG nécessite la présence en mémoire centrale de la matrice d'allocation de ressources de taille Km. LA, par opposition, peut fonctionner à partir de la valeur courante d'un flot et du flot total sur chaque arc. Cela permet d'envisager la résolution de grands problèmes sur une machine disposant d'une mémoire très modeste.

LA semble donc la mieux adaptée à la résolution de grands problèmes Construite autour de la règle d'ajustement des multiplicateurs de Van der Straatten-Tilquin et des algorithmes de direction de descente de Nguyen, l'implantation de LA promet des résultats très intéressants.

CHAPITRE TROISIEME

UN ALGORITHME POUR LA RESOLUTION DU PROBLEME DE MULTIFLOT COMPATIBLE A COUT MINIMUM

Dans le second chapitre, nous avons présenté tout un arsenal de techniques pour résoudre le modèle MCM. Nous avons analysé les propriétés théoriques et une partie de l'expérience numérique. La méthode du lagrangien augmenté nous est apparue comme la plus intéressante. Nous avons donc réalisé une implantation aussi efficace que possible de cette méthode.

L'algorithme a deux composantes essentielles. La première est la règle d'ajustement des multiplicateurs, la seconde, un algorithme pour la résolution de problèmes de flots à coûts convexes.

Nous avons choisi, pour l'ajustement des multiplicateurs, la règle de Van der Straatten-Tilquin [24]. Cette règle est bien adaptée à la structure du problème.

Parmi les méthodes proposées par Nguyen [15] pour le modèle de flot d'équilibre, nous avons choisi la méthode du simplexe convexe. Cet algorithme semble plus efficace que la méthode de Frank-Wolfe.

L'algorithme proposé ici a été codé et testé dans le but de vérifier sa viabilité. Nous voulons vérifier si l'algorithme permet de résoudre des problèmes de grande taille à un coût acceptable et observer son comportement dans différentes conditions.

Par "résoudre", nous entendons la construction d'une solution presque réalisable et presqu'optimale. La notion de coût de calcul devra être mis en relation avec la taille des problèmes traités.

3.1 L'ajustement des multiplicateurs

3.1.1 Pondération des contraintes de capacité

Nous introduisons d'abord une légère modification aux constructions de la section 2.1.6. La fonction L prendra la forme (Supposons $h_a > 0$ $\forall\ a \in A$),

$$L_u(x, \lambda) = \sum_{k \in K} c^k x^k + \frac{1}{2u} \sum_{a \in A} [\max^2\{0, \lambda_a + u \left(\frac{\sum_{k \in K} x_a^K - h_a}{h_a} \right)\} - \lambda_a^2] \quad . \qquad (3.01)$$

L'algorithme présenté est modifié en conséquence comme on le verra sous peu. Cette nouvelle fonction pondère les contraintes de capacité et réduit de ce fait les problèmes d'ajustement du critère d'arrêt.

3.1.2 L'algorithme

Au niveau supérieur du processus, on trouve l'algorithme d'ajustement des multiplicateurs et de choix des sous-problèmes.

ETAPE 0 - Initialisation

Choisir $\lambda \geq 0$, poser $L = K$, $u > 0$.

Choisir $x = (x^k)_{k \in K}$ solution réalisable de MEC.

ETAPE 1 - Choix d'un sous-problème

Si $L = \emptyset$, alors on a terminé

sinon, on choisit $\overline{k} \in L$.

ETAPE 2 - Résolution du sous-problème

Calculer $y^{\overline{k}}$ solution optimale de

$$\text{MEC}(\overline{k}) \quad \min L_u(x, \lambda)$$

$$A x^{\overline{k}} = b^{\overline{k}}$$

$$0 \leq x^{\overline{k}} \leq M$$

$$x^k \text{ fixé si } k \neq \overline{k}, \ \forall \ k \in K .$$

Poser (MODIFIE) = $(y^{\overline{k}} \neq x^{\overline{k}})$.

Poser $x^{\overline{k}} \leftarrow y^{\overline{k}}$

ETAPE 3 - Ajustement des multiplicateurs

$$\text{si } \left[\sum_{a \in A} \max^2 \{0 , u \left\{ \frac{\sum_{k \in K} x_a^k - h_a}{h_a} \right\} \right]^{1/2} < \varepsilon ,$$

alors on passe à l'étape 4.

Sinon, on pose $L = K$ et

$$\lambda_a \leftarrow \max \{0, \lambda_a + \frac{u}{K} \quad \frac{\Sigma x_a^k - h_a}{h_a} \}$$

et retour à l'étape 1.

ETAPE 4 - Révision de la liste

si (MODIFIE)

alors $L = K$

sinon $L = L - \{ \overline{k} \}$

aller à l'étape 1.

FIN

Les multiplicateurs sont donc modifiés à chaque fois qu'un sous-problème de flot est résolu. L'avantage de cette formule est de fournir un ajustement grossier des multiplicateurs plus rapidement.

3.1.3 Solution initiale

La détermination d'une solution initiale est un problème facile. Négligeant les contraintes de capacité, les flots initiaux correspondent à une affectation tout-ou-rien des demandes à coût minimum. Cela correspond à résoudre pour chaque $k \in K$ un problème de plus court chemin.

3.1.4 Critère d'arrêt

L'algorithme se termine lorsque la liste L est vide à l'étape 1. Cependant, la liste L ne peut se vider que si le critère de l'étape 3 est satisfait. Ce critère de l'étape 3 assure une solution "presque réalisable".

De plus, si la liste L est vide, la solution $(x^k)_{k \in K}$ est "presque optimale" pour les valeurs λ courantes. C'est le résultat cherché.

3.1.5 Sélection de $\overline{k}$

L'algorithme proposé nécessite le maintien des flots $(x^k)_{k \in K}$. Vu la taille des problèmes traités, ces informations sont conservées en mémoire secondaire. Lorsque $\overline{k}$ est choisi à l'étape 1, la valeur du flot $x^{\overline{k}}$ est copiée en mémoire centrale et à la fin de l'étape, la nouvelle valeur est copiée en mémoire secondaire.

Le traffic d'entrée/sortie étant très lent, le choix de $\overline{k}$ doit être fait de façon "intelligente".

Les implantations classiques traitent les flots de façon circulaire, donc dans un ordre immuable. On choisit $\overline{k}$ selon la formule:

$$\overline{k} \leftarrow \begin{cases} \overline{k} + 1 & \text{si} \quad \overline{k} < K \\ 1 & \text{si} \quad \overline{k} = K \end{cases}$$

Le maintien des valeurs des flots en mémoire secondaire peut alors être effectué à l'aide de deux fichiers séquentiels. Sur le premier, on lit les anciennes valeurs des flots et on écrit les nouvelles valeurs sur le second. Lorsque les K flots ont été parcourus, le second fichier porte une solution complète. Les deux fichiers échangent alors leurs rôles, la nouvelle solution devenant l'ancienne.

L'avantage principal de cette méthode est de permettre un certain parallélisme entre le traitement en unité centrale et les opérations d'entrée/sortie. Cependant, on observe que de nombreuses itérations se terminent alors sans modification de flot. Ces itérations sont totalement inutiles.

Nous avons tenté de réduire substantiellement ce gaspillage en examinant les flots de façon non-circulaire. Toute tentative pour implanter une telle approche est incompatible avec l'usage de fichiers séquentiels.

Nous avons donc utilisé un fichier à accès aléatoire pour stocker la solution courante. Chaque enregistrement, adressable par une clé, représente un flot x^k et est accessible directement en lecture et en écriture. Il y a donc K enregistrements.

Notons immédiatement deux caractéristiques de ce type de fichier. Un fichier à accès aléatoire ne permet pas de parallélisme entre traitement et entrée/sortie. En effet, ne pouvant prévoir la prochaine requête, on ne saurait que mettre dans le tampon. D'un autre côté, lorsqu'un vecteur de flot n'a pas été modifié, il n'est pas nécessaire de le recopier sur le fichier.

On peut penser à de nombreuses façons de choisir le prochain flot $\overline{k}$. Le but d'une règle de choix est de tenter de prévoir quels flots sont susceptibles d'être modifiés à partir des informations disponibles.

Après plusieurs tentatives, nous avons choisi une règle simple. Pour les fins de la discussion, nous dirons qu'un flot $\overline{k}$ est non-modifié si la valeur $x^{\overline{k}}$ courante au début de l'itération 2 s'avère optimale pour MEC($\overline{k}$) (pour les tolérantes courantes). Un flot $\overline{k}$ est dit optimal s'il est solution optimale pour MEC($\overline{k}$).

Notre règle est basée sur des observations empiriques lorsque la solution courante en x approche l'optimalité. A ce moment, la résolution de MEC($\overline{k}$) à l'étape 2 ne provoque de changement de flot que sur quelques arcs

"congestionnés". Seuls les flots ayant une valeur non-nulle sur ces mêmes arcs sont susceptibles de devenir non-optimals. Typiquement, ces flots sont en petit nombre.

L'idée de base de notre règle est de concentrer l'effort sur les flots qui ont été modifiés lors de leur dernier traitement. La méthode utilise deux files notées A et B. La file A contient les flots non-modifiés à leur dernier passage. La file B contient les flots modifiés à leur dernier passage.

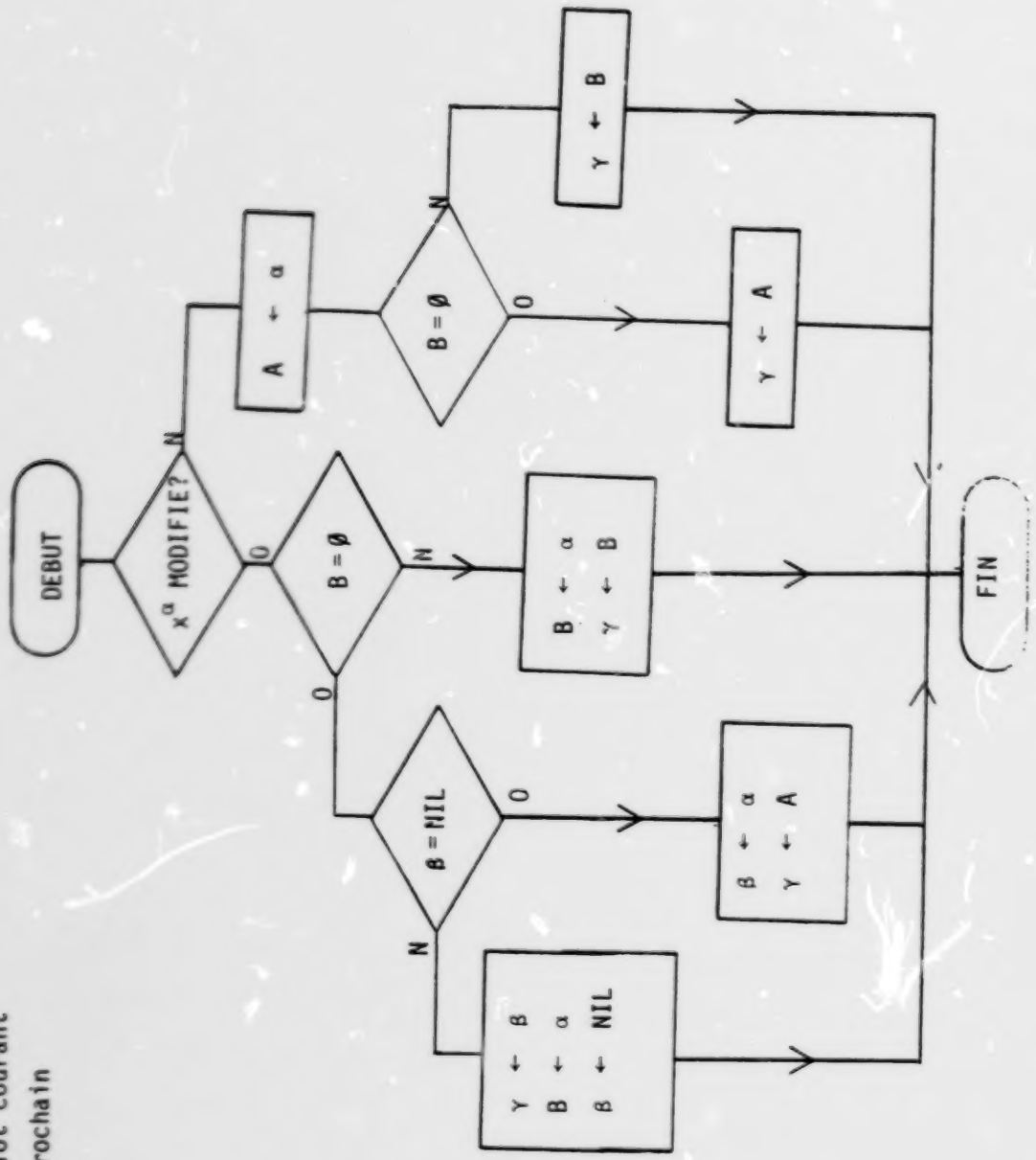
Notons,

$F \leftarrow k$	l'insertion de k dans la file F
$k \leftarrow F$	l'extraction de k de la file F
$F = \emptyset$	si F est vide

Soit	α	le flot courant
	γ	le flot à déterminer

A, B, β deux files et un scalaire décrivant l'état de la structure.

On trouve à la figure (3.1) l'organigramme de l'algorithme de sélection. Le processus est amorcé en plaçant tous les flots $k \in K$ dans la liste B et en posant β = NIL.



α est le flot courant
γ est le prochain

FIGURE 3.1
Choix du prochain flot

L'essentiel de l'algorithme consiste à choisir γ dans la file B dès que possible. Si le flot α a été modifié et que B est vide, α est mémorisé dans β pendant que défile les flots de la file A. Si un flot α de A s'avère modifié, α est inséré dans B et le prochain flot examiné est β. (qui est réinitialisé).

Cet algorithme non-seulement sélectionne les flots "intéressants", mais les ordonne dans le meilleur ordre possible.

3.2 Le simplexe convexe

A l'étape 2 de l'algorithme présenté à la section 3.1.2, nous devons résoudre un problème de flot à coût convexe. On voit facilement que la résolution de ce problème est la partie la plus coûteuse de l'algorithme (si l'on néglige le traffic E/S). Les autres étapes de l'algorithme n'impliquent en effet que quelques m opérations.

L'efficacité de l'algorithme global dépend donc essentiellement de la qualité de l'algorithme de flot à coût convexe et de son implantation. Nous avons tenté de réaliser une implantation efficace de l'algorithme de Nguyen décrit à la section 2.1.7.

L'algorithme a été développé autour de la structure de donnée de RNET, l'un des meilleurs codes de simplexe sur le réseau développé par Grigoriadis et Hsu [5] pour l'université de Rutger. Cette structure permet

de représenter le réseau et la base de façon très compacte, tout en permettant d'exécuter rapidement les opérations nécessaires à notre algorithme. Le pivotage est réalisé en quelques n opérations et seuls les arcs d'un cycle fondamental, directement retracés, sont impliqués dans le problème unidimensionnel.

Le coût de chaque itération n'est donc pas relié au nombre d'arcs, mais est plutôt de l'ordre de quelques n opérations.

Nous allons d'abord décrire la structure de donnée qui comporte deux composantes essentielles: la description du réseau et la description de la base.

3.2.1 La description du réseau

Chacun des m arcs est représenté par une entrée dans une série de vecteur décrivant ses caractéristiques.

Un arc est d'abord identifié par ses extrémités. Cela est réalisé à l'aide de deux vecteurs:

FROM(a) est le noeud d'origine de l'arc a

TO(a) est le noeud de destination de l'arc a

Pour appliquer l'algorithme, il nous faut plus d'information sur la structure du réseau. La détermination de l'arc entrant lors d'un changement de base nécessite l'examen de tous les arcs entrant en un noeud donné.

Un vecteur de m entrées permet de chaîner en liste les arcs entrant en un noeud. Un vecteur de n entrées permet d'accéder au premier arc de la liste. Nous avons:

BNEXT(a) est un arc de même destination que a (la valeur NIL indique la fin de la liste).

BFIRST(i) est le premier arc de la liste.

Nous devons aussi stocker les paramètres du problème soit:

CK(a) est le coût de l'arc pour le flot $\overline{k}$, soit $c_a^{\overline{k}}$.

H(a) est la capacité de l'arc h_a .

Pour décrire la solution courante, il faut connaître le flot $x_a^{\overline{k}}$, le flot total sur l'arc et la valeur des multiplicateurs.

XK(a) est la valeur du flot $\overline{k}$ sur l'arc a .

X(a) est le flot total sur l'arc a .

Y(a) est la valeur du multiplicateur pour l'arc a.

Pour limiter le nombre d'évaluation de la différentielle, on utilise un autre vecteur pour conserver la valeur de la dérivée première de la fonction économique

DELCK(a) est la valeur $\dfrac{\partial L_u(x, \lambda)}{\partial x_a^{\overline{k}}}$

Un dernier vecteur permet de définir des arcs de divers "types" pour, par exemple, "fermer" un arc ou annuler son coût

LTYP(n) est le type de l'arc a (valeur numérique).

A la figure (3.2), nous illustrons par un exemple la structure décrivant le réseau.

3.2.2 Structure de base et opérations sur la base

3.2.2.1 Structure de pré-ordre

Supposons d'abord que le réseau G =(N,A) est connexe, nous verrons plus loin comment lever cette hypothèse.

La matrice d'incidence A est alors de rang n-1. Il faut donc retirer une ligne de la matrice A pour éliminer la redondance. Une base B de cette nouvelle matrice A correspond à un sous-arbre, i.e. de façon équivalente:

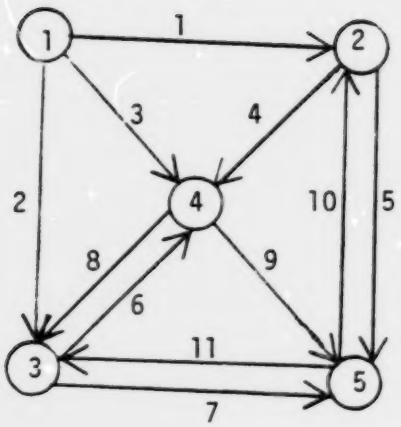
i) B compte (n-1) arcs et ne contient aucun cycle.

ii) Dans B, il existe un et un seul chemin entre chaque paire de sommets distincts.

On remarque que pour chaque arc $(i,j) \in A-B$, le chemin de i à j dans B et l'arc (i,j) forme un cycle élémentaire. Ce cycle est appelé cycle fondamental associé à (i,j) pour la base B.

Si nous choisissons arbitrairement un noeud s dit racine, la base B induit une relation de pré-ordre $\equiv$ sur l'ensemble des noeuds, soit:

$i \equiv j$ ssi i est sur le chemin de j à s.



a	FROM(a)	TO(a)	BNEXT(a)
1	1	2	10
2	1	3	8
3	1	4	4
4	2	4	6
5	2	5	7
6	3	4	NIL
7	3	5	9
8	4	3	11
9	4	5	NIL
10	5	2	NIL
11	5	3	NIL

i	BFIRST(i)
1	NIL
2	1
3	2
4	3
5	5

FIGURE 3.2

Représentation du réseau: exemple

La structure de pré-ordre est décrite par trois vecteurs de n entrées:

IRT est la source s.

ARC(i) est l'indice du premier arc sur le chemin de i à s dans B.

F(i) est le premier noeud sur le chemin de i à s.

D(i) est le nombre d'arcsdu chemin de i à s dans B.

3.2.2.2 Traçage de cycle

Cette structure permet de retracer directement les arcs du cycle fondamental associé à un arc (i,j) pour la base B et d'effectuer diverses opérations sur les arcs du cycle. Cette propriété est très importante. En effet, une fois localisé l'arc (i,j) où il apparaît intéressant de modifier le flot, l'algorithme ne travaille que sur les arcs du cycle fondamental. Hors,ce cycle est typiquement un petit sous-ensemble des m arcs de A.

D'un noeud i donné, on "remonte" le chemin vers s selon la suite de noeuds

$$i, F(i), F(F(i)),...$$

Retraçant ainsi les arcs du chemin

$$ARC(i), ARC(F(i)), ARC(F(F(i)))...$$

Les chemins originant de i et j se dirigeant vers s, ils doivent se confondre à partir d'un noeud t (possiblement t = s). Le cycle fondamental est alors formé de l'arc (i,j), du chemin de i à t et du chemin de j à t.

Progressant simultanément le long des deux chemins, on peut identifier le sommet t. Il suffit de conserver les "profondeurs" D(.) égales dans chacun des chemins. De cette façon le sommet t est atteint au même moment dans chacun des chemins.

Pour exemple, on trouvera à la figure 3.4 l'organigramme de l'algorithme permettant la modification des valeurs de flot. Toutes les opérations sur les cycles sont réalisées selon le même schéma. A la figure 3.5, nous présentons un exemple graphique.

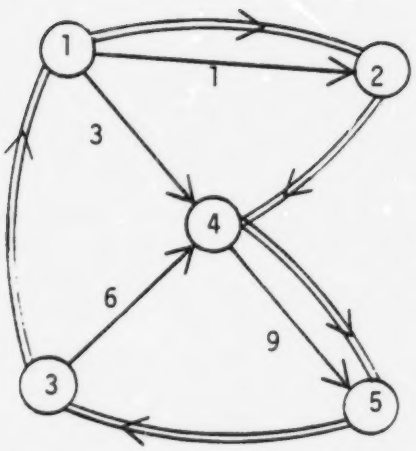
3.2.2.3 Fonction de traverse en pré-ordre

Il est facile d'étendre la relation d'ordre partiel $\equiv$ à une relation d'ordre total $|$ sur N. Nous recherchons une relation satisfaisant

$$i \equiv j \quad \Rightarrow \quad i \mid j \qquad i, j \in N \tag{3.02}$$

Nous exigerons aussi de cette relation qu'elle satisfasse la condition

$$i \mid j \mid \ell \ , \ i \equiv \ell \quad \Rightarrow \quad i \equiv j \tag{3.03}$$



i	ARC(i)	F(i)	D(i)	P(i)	R(i)
1	NIL	NIL	0	2	3
2	1	1	1	4	1
3	6	4	2	1	5
4	3	1	1	5	2
5	9	4	2	3	4

FIGURE 3.3

Représentation de la base: exemple

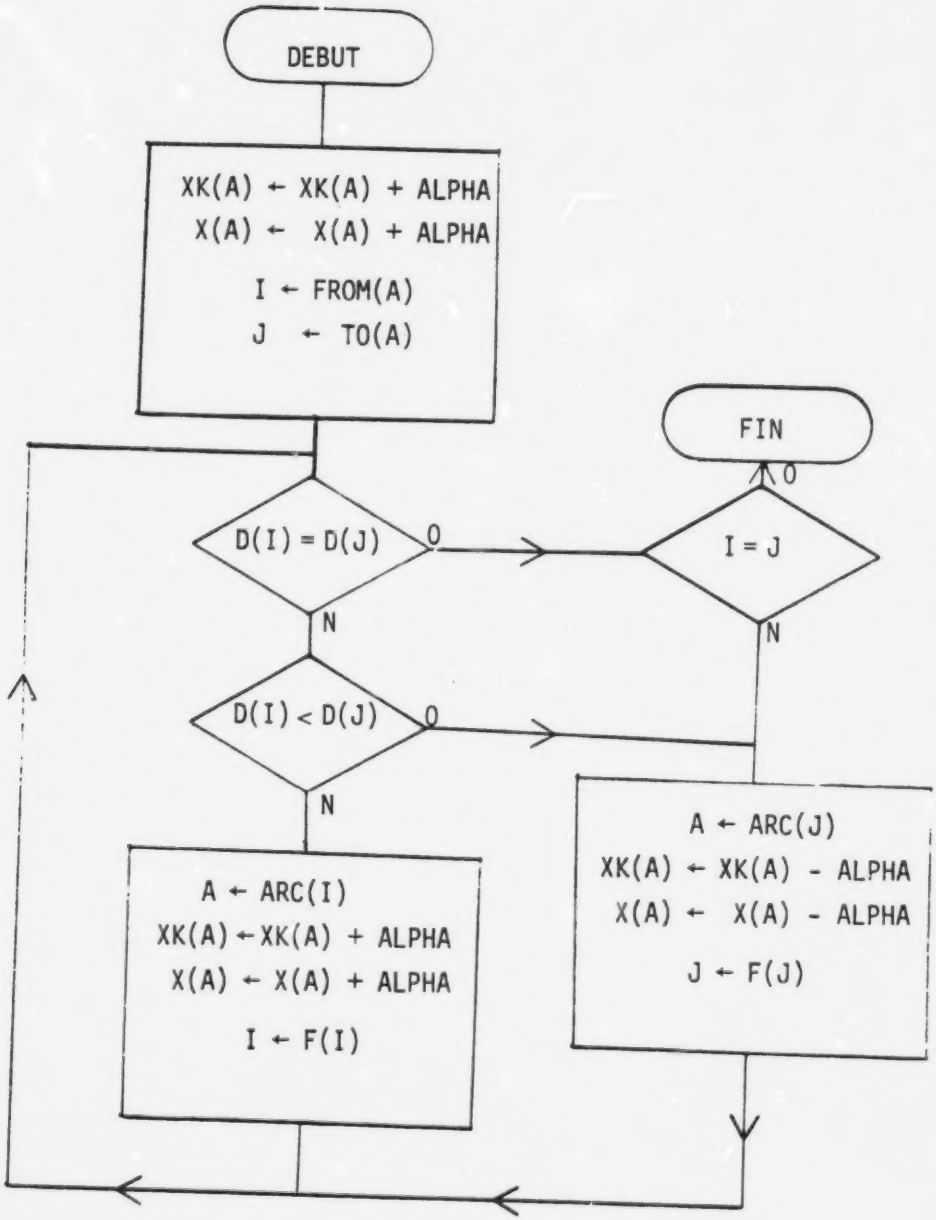
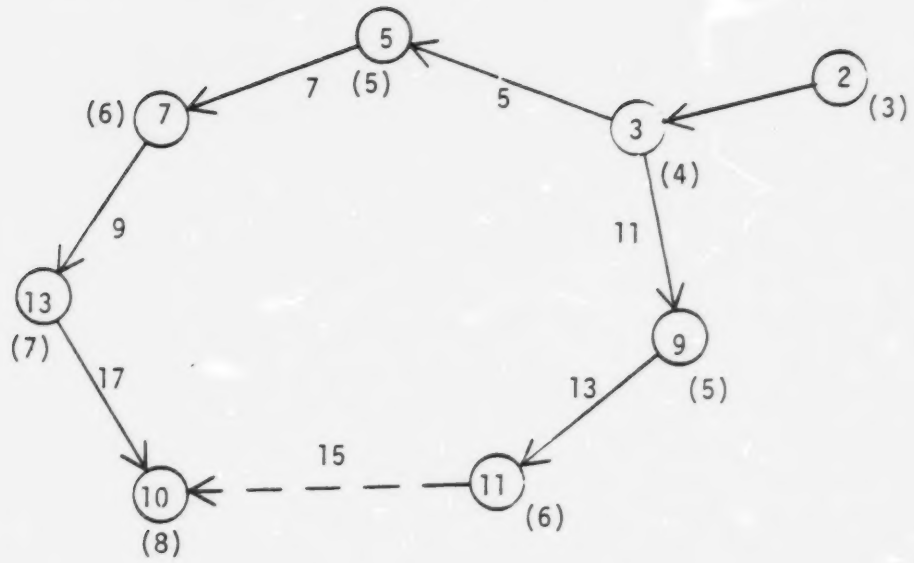


FIGURE 3.4

Modification du flot sur l'arc A: organigramme



Structure de pré-ordre

i	ARC(I)	F(I)	D(i)
2	-	-	3
3	3	2	4
5	5	3	5
7	7	5	6
9	11	3	5
10	17	13	8
11	13	9	6
13	9	7	7

Séquence de traitement des arcs

A	I	J
15	11	10
17		13
9		7
7		5
13	9	
5		3
11	3	

FIGURE 3.5

Traçage de cycle: exemple graphique

Nous introduisons plus loin une façon de construire un tel ordre total, prouvant ainsi son existence. Notons qu'il existe généralement plusieurs relations satisfaisant (3.02) et (3.03).

On peut décrire l'ordre total comme un ordonnancement $\{ i_\ell \}_{\ell=1}^{n} = N$ des sommets

où

$$i_1 \mid i_2 \ldots\ldots \quad i_{n-1} \mid i_n$$

$$i_1 = s \quad \text{et} \quad i_p = i_q \Rightarrow p = q$$

Pour fin de calcul, la façon la plus efficace de représenter la relation consiste à chaîner en liste les noeuds. Chaque noeud pointe sur le noeud qui le suit et sur celui qui le précède dans l'ordonnancement. Cela est réalisé à l'aide de deux vecteurs ayant n entrées:

$$P(i_\ell) = i_{\ell+1} \quad \text{si} \quad 1 \le \ell < n$$

$$= i_1 \quad \text{si} \quad \ell = n$$

$$R(i_\ell) \quad i_{\ell-1} \quad \text{si} \quad 1 < \ell \le n$$

$$i_n \quad \text{si} \quad \ell = 1$$

On voit bien que

$$P(R(i)) = R(P(i)) = i \qquad i \in N$$

La fonction qui associe à chaque noeud le noeud suivant (le vecteur P) est appelée fonction de traverse en pré-ordre. La figure 3.2 est un exemple de la structure décrivant la base.

3.2.2.4 Calcul des potentiels

La fonction de traverse en pré-ordre permet de résoudre efficacement le système d'équation (2.02-03).

Nous introduisons d'abord le concept de sous-arbre N(t) d'un noeud t. C'est l'ensemble des noeuds i pour lesquels le chemin de i à s passe par t

$$N(t) = \{ i \in N \mid t \in i \} .$$

Si nous parcourons les noeuds, commençant en s, selon la fonction de traverse en pré-ordre, nous sommes assurés qu'en parvenant au sommet i, tous les noeuds sur le chemin de s à i ont été parcourus. Cela est assuré par la propriété (3.02). Nous pouvons alors calculer, de façon inductive , en i donné:

$$U(i) = U(F(i)) \mp DELCK(ARC(i)) .$$

Le vecteur U à n entrées permet de stocker les valeurs π_i .

La propriété (3.03) nous assure que les noeuds du sous-arbre N(t) suivent immédiatement t dans la fonction de traverse en pré-ordre. De plus, il est clair que:

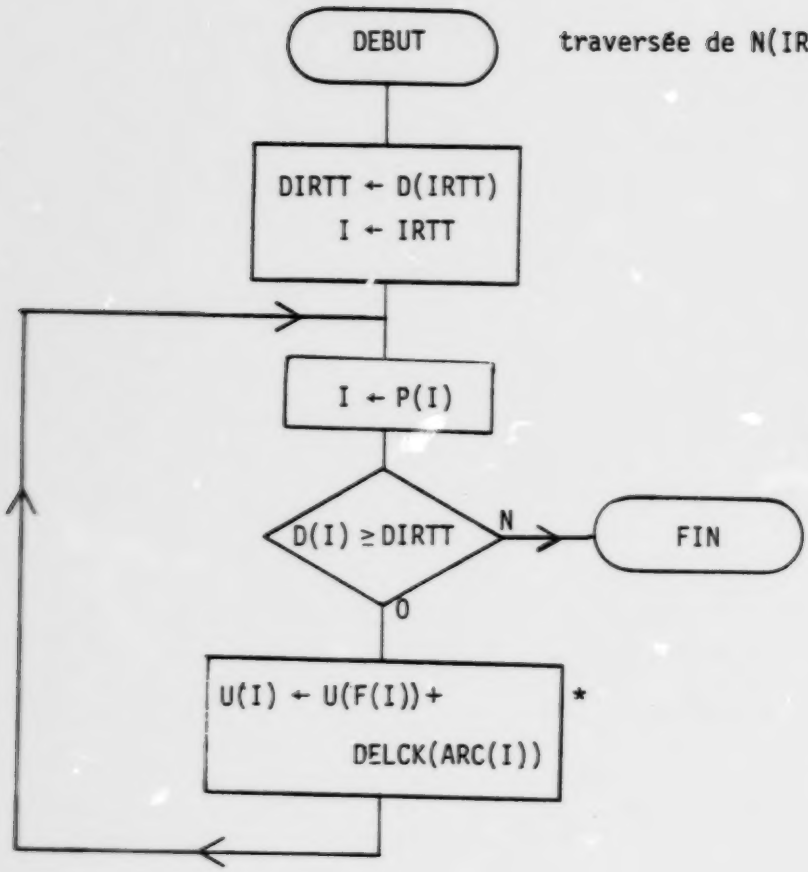
$$i \in N(t) \rightarrow D(i) > D(t)$$

On en conclut que les noeuds de N(i) sont les noeuds suivant t de "profondeur" plus grande que t. Cette caractérisation du sous-arbre N(.) par rapport à la fonction de traverse permet de limiter les calculs au sous-arbre lorsque nécessaire. L'algorithme permettant la mise-à-jour des potentiels sur un sous-arbre est présenté à la figure 3.6.

3.2.2.5 Pivotage

Nous avons défini une structure complexe pour décrire la base courante. Cette structure non seulement permet d'exécuter rapidement les opérations impliquant la base, mais peut aussi être rapidement mise-à-jour lors d'un changement de base.

L'algorithme de pivotage est la raison principale du succès de RNET et de ses concurrents. Notre algorithme est de même nature, mais présente des différences d'ordre technique par rapport à RNET. De fait, notre algorithme, pour une même structure de départ, produira une fonction de traverse en pré-ordre différente.



* sous l'hypothèse que la base est une arborescence

FIGURE 3.6

Mise-à-jour des potentiels: organigramme

L'intérêt de notre algorithme est sa relative simplicité de conception pour un nombre d'opérations très comparable. Cette simplicité permet une implantation rapide et fiable pour des applications de recherche. D'autre part, la méthode présente un grand intérêt pédagogique, car elle est facile à comprendre.

Supposons donc connue une base B, un arc sortant p et un arc entrant q possédant une extrémité commune t.

Sans perte de généralité, nous poserons,

$$p = (i,t) \qquad q = (t, j) \ .$$

La première opération consiste à mettre à jour la relation de pré-ordre. Cette opération est très facile à réaliser; il suffit de faire

$$ARC(t) \leftarrow q$$
$$F(t) \leftarrow j$$

Nous devons aussi mettre à jour la profondeur et les potentiels sur l'ensemble de noeuds

$$T = \{t\} \ U \ N(t)$$

On voit facilement que si

$$DELU = [\ U(j) \stackrel{+}{-} DELCK(p)\ ] - U(t)$$
$$DELD = [\ D(j) + 1\ ] - D(t)$$

la mise-à-jour se limite à calculer, pour tout $\ell \in T$

$$U(\ell) \leftarrow U(\ell) + DELU$$
$$D(\ell) \leftarrow D(\ell) + DELD$$

Cette opération est réalisée en traversant le sous-arbre N(t) comme lors du calcul des potentiels.

Il reste à ajuster la fonction de traverse en pré-ordre pour la nouvelle base. Notons d'abord que si i =j, alors il n'y a rien à faire, car p et q sont des arcs opposés. Nous supposerons donc que $i \neq j$.

Si ℓ est le dernier sommet de T, les sommets se présentent initialement dans l'ordre suivant, (sauf si $N(t) = \emptyset$, auquel cas $t = \ell$)

$$\ldots i \mid P(i)\ldots.R(t) \mid t \mid P(t)\ldots.\ell < P(\ell) .$$

L'ensemble T disparaissant de N(i) pour être ajouté à N(j), nous voulons obtenir la structure

$$\ldots i \mid P(i)\ldots.R(t) \mid P(\ell)\ldots.$$
$$\ldots j \mid t \mid P(t)\ldots.\ell \mid P(j)\ldots.$$

La fonction de traverse en pré-ordre est donc rapidement mise-à-jour en posant

$$P(R(t)) \leftarrow P(\ell)$$
$$P(\ell) \leftarrow P(j)$$
$$P(j) \leftarrow t$$

Toutes ces notions sont intégrées dans l'algorithme de micro-pivot dont l'organigramme est présenté à la figure 3.7.

Le micro-pivot permet d'effectuer un pivot lorsque l'arc entrant et l'arc sortant ont une extrémité commune. Il reste à montrer comment utiliser le micro-pivot pour pivoter dans le cas général. Considérons donc une base B, un arc sortant p et un arc entrant q.

Si $[B-\{p\}] \cup \{q\}$ est une base,

alors p est un arc du cycle fondamental associé à l'arc q et à la base B. En effet, si p n'est pas du cycle fondamental, la nouvelle base contiendrait un cycle.

Soit t, la racine du cycle fondamental, et supposons, sans perte de généralité, que l'arc p est sur le chemin de j(p) à t. Définissons la suite $\{a_\ell\}_{\ell=0}^{L}$ des arcs du chemin:

$$a_0 = q$$
$$a_1 = ARC(j(p))$$
$$a_2 = ARC(F(j(p))$$
$$\cdots$$
$$a_L = p$$

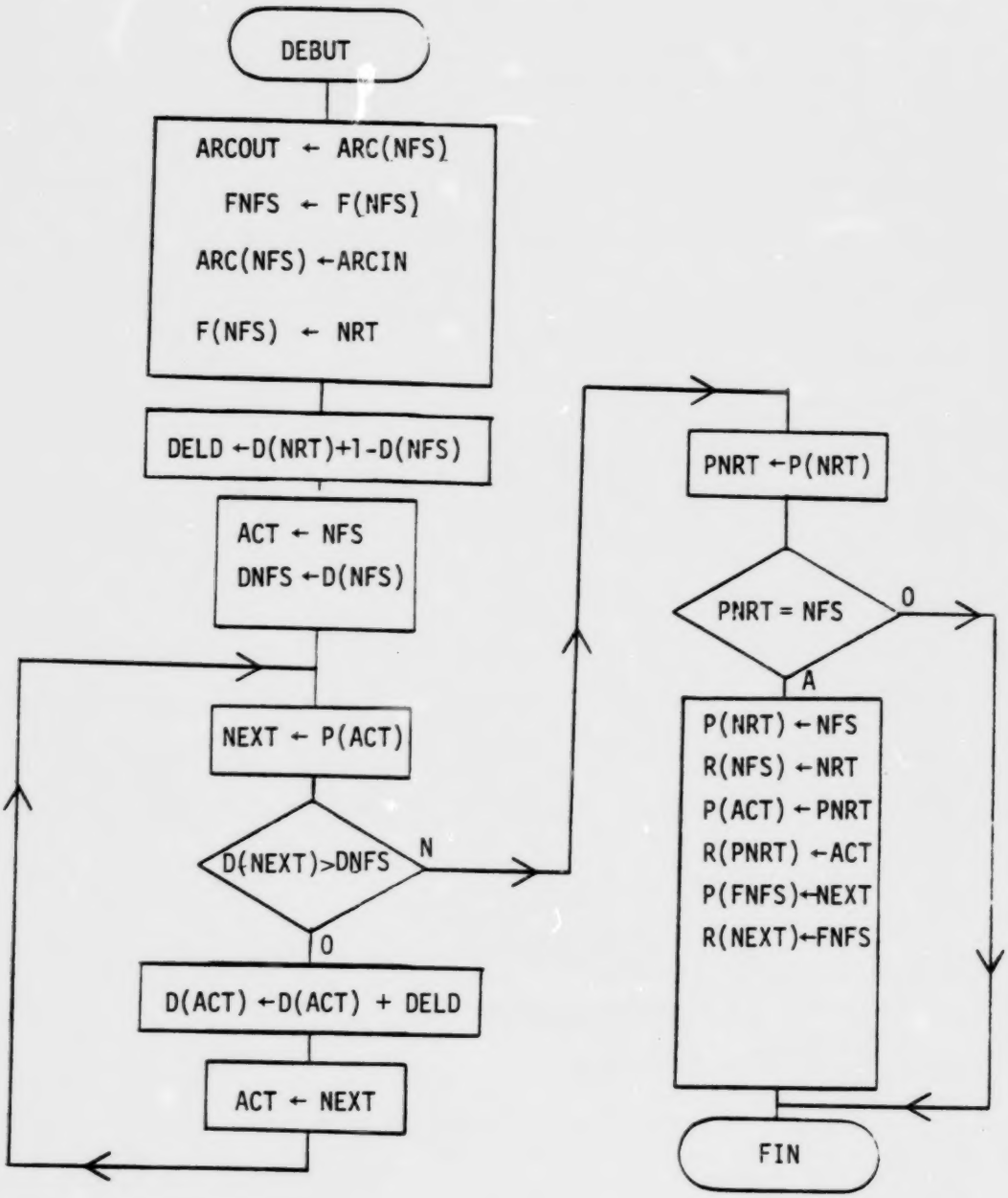


FIGURE 3.7

Micro-pivot: organigramme

Le pivot peut alors être réalisé en effectuant L micro-pivots. Le ℓ ième micro-pivot consistant à entrer l'arc $a_{\ell-1}$ pour sortir l'arc a_ℓ. Pour mettre en évidence la consistance de la méthode, il suffit de noter que $a_{\ell-1}$ et a_ℓ ont une extrémité commune pour $1 \leq \ell \leq L$ et que

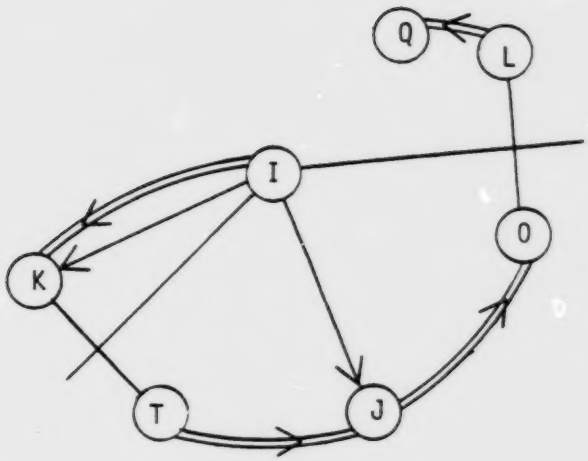
$$[..[[B \cup \{a_0\}] - \{a_1\} \cup \{a_1\}] - \{a_2\} \ldots] - \{a_L\}$$

$$= [B - \cup \{a_0\}] - \{a_L\}$$

$$= [B \cup \{q\}] - \{p\} .$$

De fait, les L micro-pivots successifs sont réalisés en chaîne, chaque micro-pivot utilisant l'information disponible à la fin du précédent. Le résultat final est un algorithme d'une efficacité très comparable à l'algorithme de pivot de RNET. Les deux algorithmes partagent en effet les mêmes propriétés: Seuls les noeuds concernés sont parcourus et chaque noeud n'est traversé qu'une seule fois.

A la figure (3.8), nous présentons la notation graphique de M.-A. Denis pour décrire la structure de base. Les figures (3.9) à (3.11) sont un exemple de l'application de l'algorithme nécessitant deux micro-pivots. La figure (3.12) présente le résultat de l'algorithme de pivotage de RNET appliqué à la situation initiale de la figure (3.8). On remarquera la différence entre les fonctions de traverse produites par les deux algorithmes.

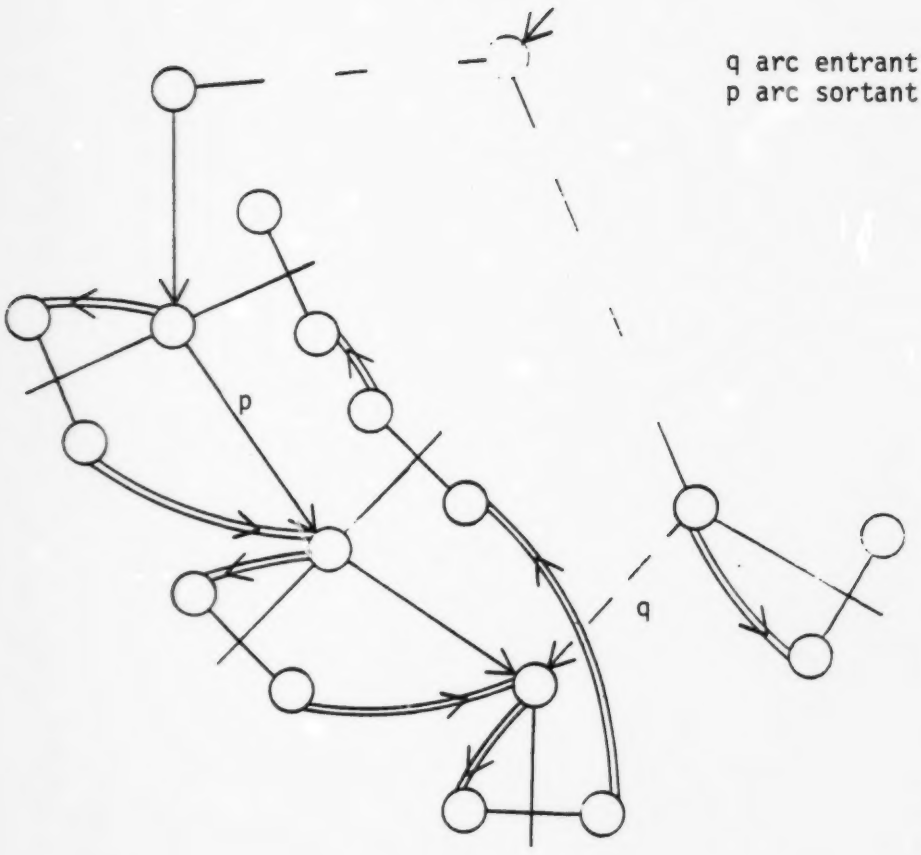


Le graphe correspond à la structure suivante

F(K) = I	P(T) = J	P(L)= Q
F(J) = I	R(J) = T	R(Q) = L
P(I) = K	P(J) = 0	L ∈ N(I)
R(K) = I	R(0) = J	Q ∉ N(I)

FIGURE 3.8

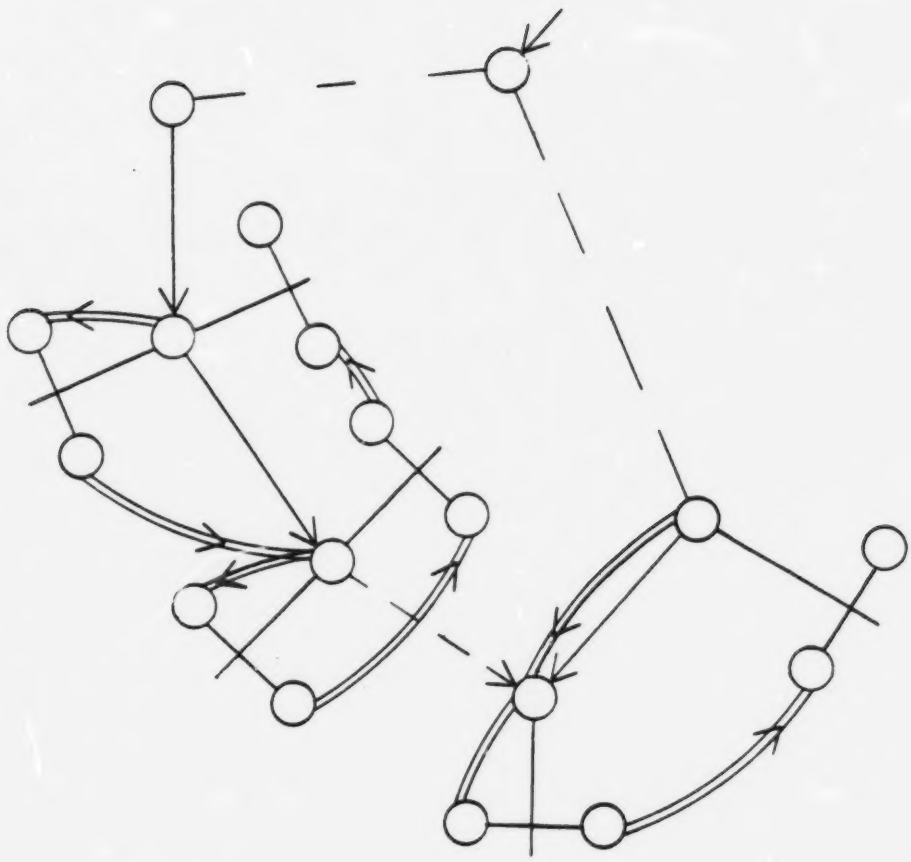
Notation graphique de M.-A. Denis: exemple



SITUATION INITIALE

FIGURE 3.9

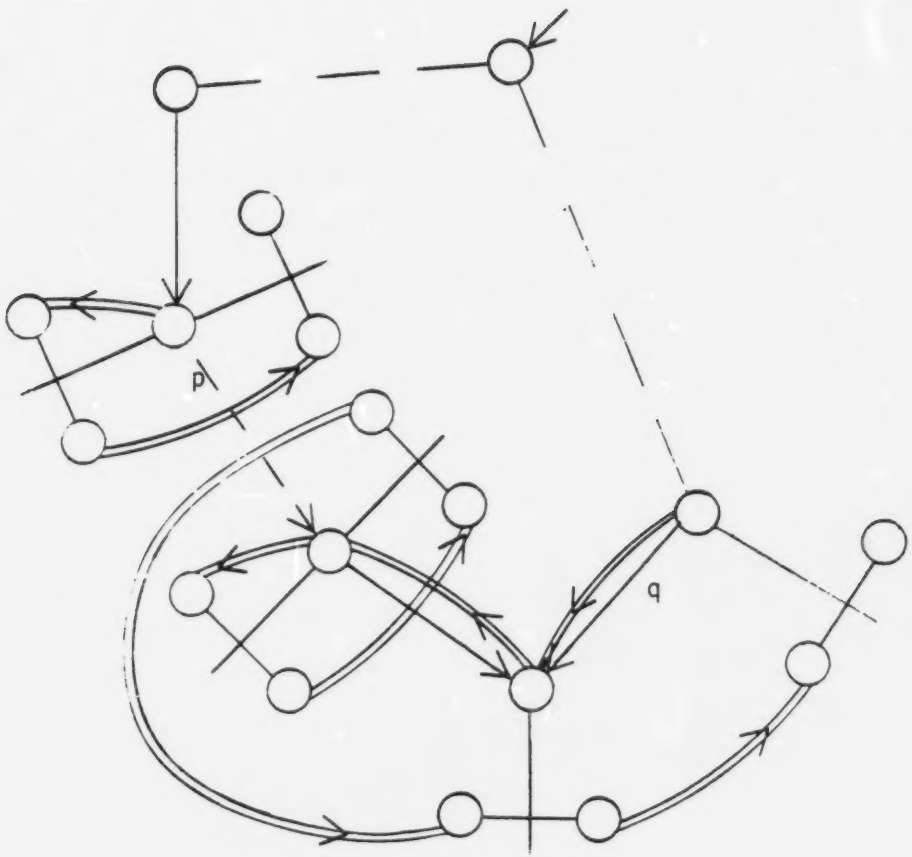
Méthode de pivotage: exemple graphique



APRES 1 MICRO-PIVOT

FIGURE 3.10

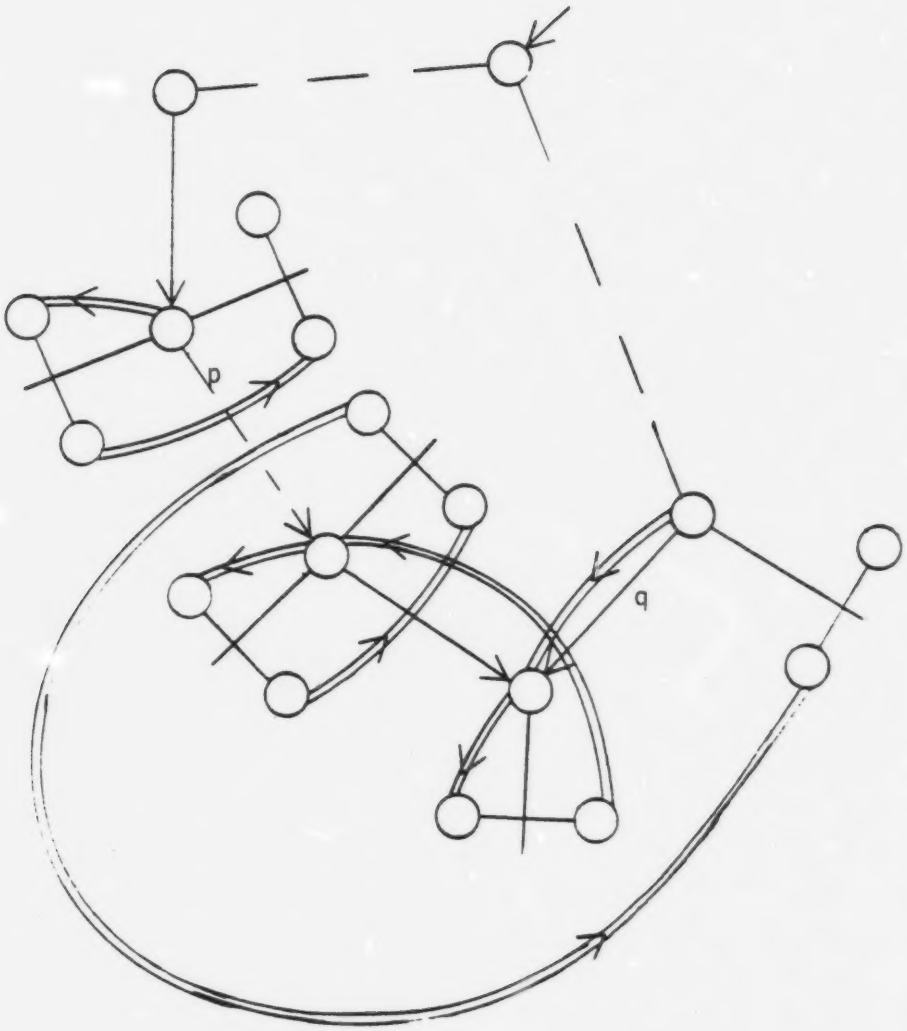
Méthode de pivotage: exemple graphique



APRES 2 MICRO-PIVOTS

FIGURE 3.11

Méthode de pivotage: exemple graphique



APRES LE PIVOT DE RNET

FIGURE 3.12

Méthode de pivotage: exemple graphique

Nous noterons en terminant que nous avons volontairement négligé les détails concernant l'orientation des arcs pour alléger la discussion.

3.2.2.6 Arc artificiel

Rappelons que toutes les bases considérées par l'algorithme sont des arborescences. Hors, typiquement, il n'existe pas de chemin orienté entre chaque paire de sommets. Ce problème peut être contourné en ajoutant un arc artificiel de la source s vers chaque noeud. Affectant un très grand coût à ces arcs, ils ne portent du flot que si le problème n'est pas réalisable. Le rôle de ces arcs dans la base est joué par un seul arc, d'où gain d'espace. Cet arc peut être pivoté hors de la base au besoin, de façon transparente.

3.2.2.7 Construction de la structure de base initiale

La solution initiale de l'algorithme est déterminée en résolvant un problème de plus court chemin. L'algorithme utilisé est une variante de l'algorithme de Ford Moore et Bellman. Cet algorithme est bien connu et a été l'objet d'une abondante littérature [14 , 16 , 17]. Nous n'en décrirons pas les détails.

Cet algorithme rend comme solution une arborescence. Cette arborescence est décrite dans le vecteur ARC. On en déduit facilement le vecteur F:

$$F(i) = TO(ARC(i)) \quad \forall\ i \neq s \ .$$

Nous pouvons ensuite déterminer la solution de départ en x, le flot étant affecté sur les arcs de l'arborescence. Débutant avec un flot nul, pour chaque noeud $i \neq s$ tel que $b_i > 0$, nous retraçons le chemin de i à s, ajoutant b_i unités de flot sur chaque arc du chemin.

Nous construisons ensuite la fonction de traverse en pré-ordre selon la méthode suivante:

Soit S = {s}, il est facile d'introduire une relation d'ordre total sur un singleton,simplement:

$$P(s) = s \ .$$

Choisissons $i \notin S$ et considérons la suite $\{ i_\ell \}_{\ell=1}^{L}$ des noeuds sur le chemin de i à s:

$$i_0 = i$$
$$i_1 = F(i)$$
$$i_2 = F(F(i))$$
$$\ldots$$
$$i_L = S$$

et déterminons i_r le premier noeud du chemin de i à s qui appartient à S. Posons

$$S \leftarrow S \cup \{ i_\ell \}_{\ell=1}^{r-1} .$$

Nous mettons à jour la relation d'ordre total en insérant les noeuds du chemin de i à s après i_r, soit:

$$\ldots i_r \mid i_{r-1} \mid i_{r-2} \ldots . i_0 \mid P(i_r)$$

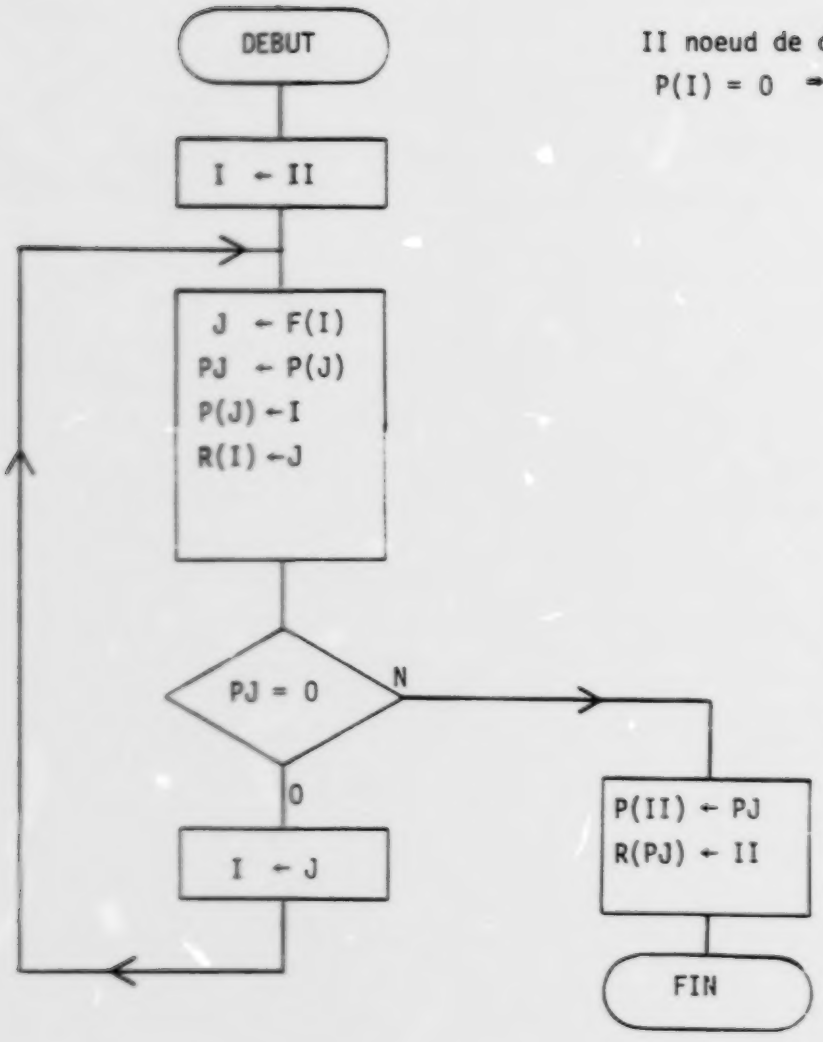
Au niveau de la fonction de traverse, il faut poser:

$$P(i_0) \leftarrow P(i_r)$$

$$P(i_\ell) \leftarrow i_{\ell-1} \qquad \forall \ 1 \leq \ell \leq r .$$

Nous présentons à la figure (3.13) l'algorithme réalisant cette opération. Au départ et à chaque fois que ce procédé est appliqué, nous avons une fonction de traverse valide pour l'ensemble S. Chaque itération ajoutant au moins un noeud à S, il suffit de répéter le procédé jusqu'à ce que S = N .

La dernière opération consiste à calculer la profondeur (le vecteur D) et les potentiels en traversant l'arbre selon la fonction de traverse que nous avons défini précédemment.



II noeud de départ II∉ S
P(I) = 0 ⇒ i ∉ S

FIGURE 3.13

Ajout d'un chemin à la fonction de
pré-ordre: organigramme

3.2.3 L'algorithme

Nous avons introduit les structures de donnée permettant de représenter le réseau et la base de façon compacte. Cette structure permet de réaliser efficacement trois opérations essentielles:

i) Le traçage d'un cycle fondamental

ii) La traversée en pré-ordre d'un sous-arbre

iii) Le pivotage

Nous allons reprendre chaque étape de l'algorithme introduit en 2.1.7.3. Nous voulons d'abord montrer comment les structures de donnée permettent une implantation efficace de l'algorithme, mais aussi indiquer certains raffinements techniques qui ont une grande influence sur la vitesse d'exécution.

3.2.3.1 Initialisation

Au début de l'algorithme, nous disposons des valeurs courantes du flot $x^{\overline{k}}$ et du flot total sur chaque arc. Nous devons déterminer la valeur de la première dérivée pour chaque arc, ce qui nécessite m évaluations de la différentielle.

La structure de base est reconstruite à partir du vecteur ARC selon la méthode décrite en 3.2.2.7. La valeur des potentiels est alors calculée de façon parallèle au calcul de la profondeur, soit en traversant l'arbre selon la fonction de traverse en pré-ordre.

3.2.3.2 Choix de la direction de descente

Nous avons pris ici beaucoup de liberté par rapport à l'algorithme déjà décrit. Telle qu'énoncée en 2.1.7.3, la détermination de la direction de descente nécessite l'examen de tous les arcs hors-base pour trouver le "meilleur" arc candidat à une modification de flot. Cet examen s'avère coûteux sur un réseau réel, comptant beaucoup d'arcs. Il s'avère aussi peu efficace car, près de l'optimum, la grande majorité des arcs satisfont le critère d'arrêt et sont traités inutilement.

Pour contourner ce problème, nous parcourons les arcs pour construire une courte liste d'arcs candidats. Nous insérons dans cette liste tous les arcs a qui ne satisfont pas le critère suivant:

$$-\varepsilon \leq \bar{c}_{N_a} \leq \varepsilon \quad \text{ou} \quad (c_{N_a} > \varepsilon \quad \text{et} \quad x^{\bar{k}}_{N_a} < \tau_0) \tag{3.04}$$

A chaque itération, nous prendrons pour construire la direction de descente le premier arc a de la liste de candidats ne satisfaisant pas le critère (3.04). Si tous les arcs de la liste de candidats satisfont le critère, nous parcourerons de nouveau tous les arcs, insérant dans la liste les arcs a ne satisfaisant pas le critère (3.04).

Notons immédiatement que si aucun arc n'est ajouté à la liste de candidats, le critère d'arrêt de l'algorithme est satisfait. La liste de candidats étant de longeur finie, les arcs excédentaires seront détruits

selon une stratégie FIFO. Si la fonction objectif est mal conditionnée, il se produit souvent un phénomène de "balancement" entre deux arcs hors-base. Modifier le flot sur l'un fait de l'autre un candidat à une modification de flot. On doit alors cycler sur ces deux arcs. Le fait de ne pas purger la liste permet à l'algorithme d'"apprendre" quels sont les arcs concernés par cet effet de balancier.

Bien entendu, nous disposons des potentiels sur les noeuds pour calculer les $\bar{c}_{N_a}$. Comme le propose Nguyen [15], nous calculerons la quantité $\bar{c}_{N_a}$ selon la formule:

$$\bar{c}_{N_a} = [\,(c_{N_a} + \pi_{i(a)}\,]\ /\ \pi_{j(a)}$$

Si l'on interprète π_ℓ comme la longueur du chemin de s à ℓ ($\pi_s = 0$) $\bar{c}_{N_a}$ est la différence entre la longueur de deux chemins de s à j pondérée par la longueur de l'un des chemins. Ce critère permet de "mettre à l'échelle" les coûts réduits et évite ainsi de faire des itérations qui n'offrent que des gains minimes. Les performances de l'algorithme en sont grandement améliorées.

3.2.3.3 Le calcul du pas maximal

Nous avons maintenant identifié un arc a où il apparaît intéressant de modifier le flot. Retraçant le cycle comme indiqué en 3.2.2.2, nous calculons facilement τ_{max} . De plus, si $\tau_{max} < \tau_0$ l'arc sortant tel que défini

en 2.1.7.3 est retracé naturellement, les chemins étant parcourus de j(a) vers t.

3.2.3.4 Recherche linéaire

Nous allons supposer que $\bar{c}_{N_a} < -\varepsilon$ et que nous désirons augmenter le flot sur l'arc a. Nous connaissons τ_{max} la quantité maximale de flot que nous pouvons ajouter pour conserver la réalisabilité.

Nous nous intéressons à la résolution du modèle MEC($\bar{k}$) tel qu'énoncé en 2.1.7.3. Selon la notation de 2.1.7.3, nous devons résoudre

$$\text{PRU} \qquad \min f(\tau)$$
$$0 \leq \tau \leq \tau_{max}$$

Nous avons défini en 2.1.7.3 la forme de la fonction f. La valeur de la dérivée première de f en τ est calculée à peu de frais en retraçant les arcs du cycle fondamental, comme indiqué en 3.2.2.2.

Nous avons implanté deux méthodes pour résoudre le problème unidimensionnel.

La première méthode alterne les itérations de "regula falsi" et de la méthode de bissection. Considérons l'algorithme suivant:

ETAPE 0 - Cas trivial

Si $f'(\tau_{max}) < 0$, $\tau^* = \tau_{max}$

et on a terminé.

ETAPE 1 - Initialisation

Poser $\tau_1 \leftarrow 0$

$\tau_2 \leftarrow \tau_{max}$.

ETAPE 2 - Choix d'un troisième point

Choisir τ_3 tel que $\tau_1 < \tau_3 < \tau_2$.

ETAPE 3 - Critère d'arrêt

Si $|f'(\tau_3)| < \varepsilon$, $\tau^* = \tau_3$

et on a terminé.

ETAPE 4 - Choix du nouvel encadrement

Si $f'(\tau_3) < 0$ alors $\tau_1 \leftarrow \tau_3$

sinon $\tau_2 \leftarrow \tau_3$

FIN

L'idée de cet algorithme est de fournir un encadrement $[\tau_1 , \tau_2]$ de plus en plus fin du zéro de f' ($f'(\tau_1) < 0 < f'(\tau_2)$).

Les méthodes de "regula falsi" et de bissection se distinguent par le choix du troisième point τ_3 à l'étape 2.

La méthode de bissection choisit simplement le point-milieu de l'intervalle soit:

$$\tau_3 = \frac{\tau_1 + \tau_2}{2}$$

La méthode de "regula falsi" traite la fonction f comme une quadratique dont on connaît la pente aux points τ_1 et τ_2. Cela revient à considérer la dérivée comme une droite dont on connaît deux points. Le zéro d'une telle droite se calcule par interpolation

$$\tau_3 = \tau_1 - \left(\frac{\tau_2 - \tau_1}{f'(\tau_2) - f'(\tau_1)} \right) f'(\tau_1)$$

Nous avons choisi d'alterner les deux méthodes, car si "regula falsi" a de meilleures propriétés de convergence, elle se comporte mal lorsque la fonction courbe fortement près d'une extrémité de l'intervalle. Les itérations de la méthode de bissection limite ce problème. (voir les figures 3.14 et 3.15). Cette stratégie est due à Guélat [7].

La seconde méthode implantée exploite la structure de la dérivée. Cette fonction s'écrit

$$f'(\tau) = \sum_{p \in A} \delta_p^a \cdot \left(c_p^{\overline{k}} + \frac{1}{h_p} \cdot \max \{0, \lambda p + u \sum_{k \in K} x_p^k - h_p + \delta_p^a \tau \} \right)$$

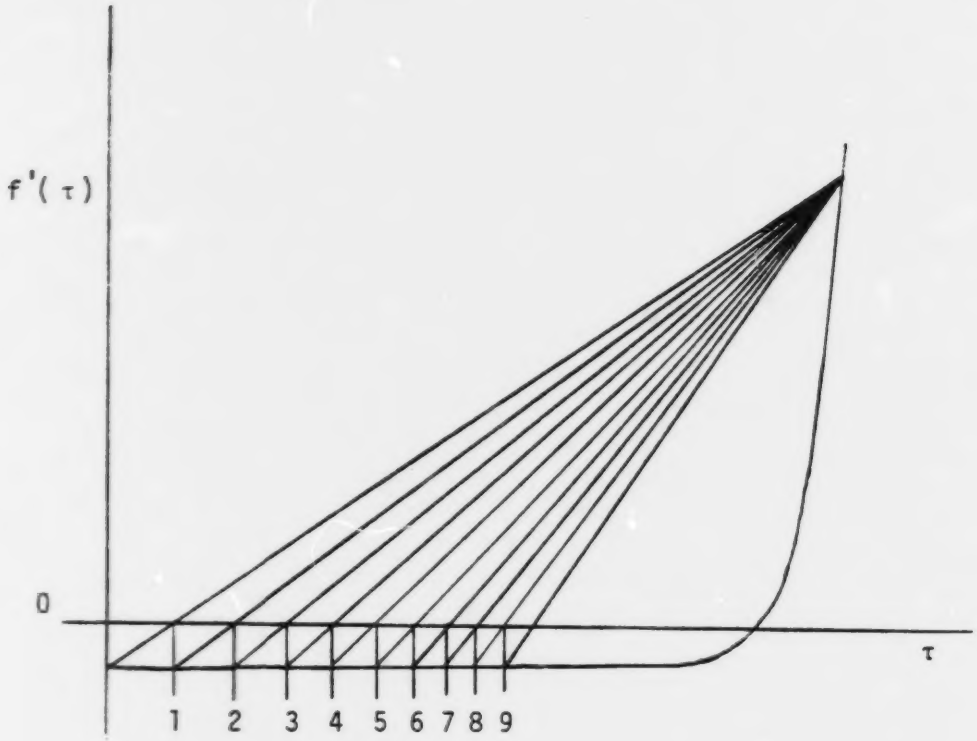
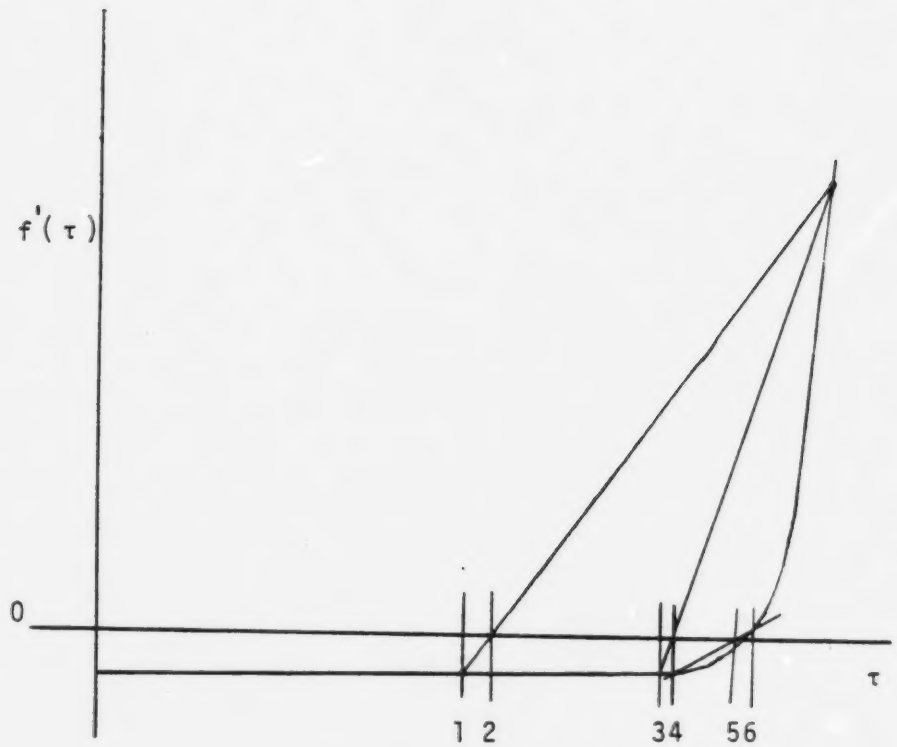


FIGURE 3.14

Convergence lente pour "regula falsi": exemple



points 1, 3 et 5 = bissection

points 2, 4 et 6 = regula falsi

FIGURE 3.15

Regula falsi + bissection: exemple

La fonction f' est continue et linéaire par morceaux. A chaque arc du cycle fondamental correspond un point d'inflexion. L'ensemble des points d'inflexion s'écrit

$$I = \{ \tau \mid \tau = -\delta_p^a \left(\sum_{K \in K} x_p^k - h_p + \frac{\lambda_p \, h_p}{u} \right) \text{ pour } \delta_p^a \neq 0 \}$$

Supposons toujours que $f'(0) < 0$ alors on calcule facilement τ^* le pas optimal selon la règle.

Si $f'(\tau_{max}) < 0$ alors $\tau^* = \tau_{max}$

sinon soit

$$\tau_1 = \max \{ \tau \mid f'(\tau) \leq 0 \text{ et } \tau \in I \cup \{ 0, \tau_{max} \} \}$$

$$\tau_2 = \min \{ \tau \mid f'(\tau) \geq 0 \text{ et } \tau \in I \cup \{ 0, \tau_{max} \} \}$$

si $\tau_1 = \tau_2$ alors $\tau^* = \tau_1$

sinon

$$\tau_* = \tau_1 - \left(\frac{\tau_2 - \tau_1}{f'(\tau_2) - f'(\tau_1)} \right) . f'(\tau_1)$$

Cela correspond à calculer le zéro en interpolant entre les deux points d'inflexion qui encadrent le zéro. Les points d'inflexion sont calculés en retraçant les arcs du cycle comme en 3.2.2.2.

On peut refaire toute l'analyse précédente, de façon symétrique, dans le cas où nous voulons diminuer le flot sur l'arc a ($\overline{c}_{N_a} > \varepsilon$).

3.2.3.4 Modification de flot

La modification des valeurs de flot est réalisée en retraçant le cycle. Au même moment, ou de façon parallèle, nous mettons à jour la valeur de la dérivée première pour les arcs du cycle. Ces valeurs sont conservées dans le vecteur DELCK.

L'itération est complétée en recalculant les valeurs des potentiels. Cela est réalisé en traversant le sous-arbre N(t) où le noeud t est la racine du cycle associé à l'arc hors-base a.

3.2.3.5 Pivotage

Nous avons longuement décrit la méthode de pivotage. Il reste à préciser comment l'arc entrant est déterminé. Les vecteurs BFIRST et BNEXT permettent de retracer directement les arcs entrant au noeud j(a). Il suffira donc d'examiner quelques arcs pour trouver notre arc entrant.

CHAPITRE QUATRIEME

RESULTATS NUMERIQUES

Nous n'avons pas trouvé dans la littérature d'expérience numérique concernant des problèmes de taille réaliste (voir 2.2.3). Il sera donc impossible de comparer les performances de notre algorithme dans le contexte que nous avons choisi.

De plus, notre décision de concentrer nos efforts sur une seule approche nous prive d'une autre source d'information. Nous nous sommes donc contentés de vérifier la viabilité de la méthode du lagrangien augmenté pour la résolution de problèmes de grande taille.

Nous avons construit nos problèmes de type MCM à partir de données déjà codifiées pour la simulation de la circulation urbaine. Il s'agit de problèmes de multiflot d'équilibre. Nous avons utilisés ces problèmes pour mettre au point l'implantation du simplexe convexe. Nous comparons nos résultats avec ceux de Guélat [7].

Nous avons ensuite construit des problèmes de type MCM en affectant des capacités sur les arcs du réseau.

4.1 Nature des problèmes

La recherche d'un banc d'essai pour notre problème s'est avérée difficile. Nous avons déjà souligné l'absence de problèmes de taille suffisante dans la littérature. Les problèmes réels dans le domaine des télécommunications, par exemple, sont en général confidentiels.

Nous avons aussi considéré la possibilité de construire des problèmes à l'aide de fonctions pseudo-aléatoires. Il nous est apparu qu'il est très difficile de générer de cette façon un problème cohérent et sans biais. La génération automatique de problèmes constitue un champ d'étude en soi et n'était pas le but de ce travail.

Nous nous sommes donc tourné du côté des problèmes d'affectation de trafic pour obtenir des problèmes de taille intéressante. Nous disposions alors de données déjà codifiées,d'où un gain de temps appréciable.

Le trafic routier urbain peut être modélisé comme un problème de multiflot d'équilibre (voir Nguyen [15]).

Dans ce contexte, le réseau représente le système routier primaire d'une région. Les noeuds correspondent à des intersections ou à des centroïdes. Un centroïde est un lieu d'où proviennent et/où se dirigent les usagers. C'est, par exemple, un quartier résidentiel, un complexe administratif ou commercial, un pont. Les arcs correspondent aux segments de route reliant deux intersections où relient les centroïdes aux intersections.

Nous connaissons le nombre d'usagers d_{ij} (demande) désirant se déplacer d'un centroïde i (origine) à un autre j (destination). Nous associons un flot k à chaque centroïde en regroupant les demandes de même origine k formant ainsi le vecteur de demande b^k

$$b_i^k = \begin{cases} \sum_{j \in K-\{k\}} d_{kj} & \text{si} \quad i = k \\ -d_{ki} & \text{si} \quad i \in K - \{k\} \\ 0 & i \in N - K \end{cases}$$

où K est l'ensemble des centroïdes et N l'ensemble des noeuds. Le flot x_a^k représente le nombre de véhicules (ou usagers) venant de k en passant par l'arc a.

Il est raisonnable de postuler que le temps nécessaire pour parcourir un segment est fonction du nombre total de véhicules v_a présents sur le segment. Typiquement, le segment est parcouru à la vitesse maximale permise lorsque quelques véhicules sont présents et à une vitesse presque nulle lorsqu'un grand nombre de véhicules provoque un embouteillage. Le temps de traversée du segment a est approximé par la fonction

$$s_a(v) = t_a^0 \left(1 + \alpha_a (v_a/h_a)^{\beta a}\right)$$

$$= t_a^0 + t_a^0 \alpha_a (v_a/h_a)^{\beta a} \qquad (4.1)$$

où t_a^0 est le temps de traversée à vide, lié à la longueur du segment et à la vitesse permise. Les paramètres α_a, β_a et h_a sont liés aux caractéristiques du lien (nombre de voies, signalisation, etc...).

Le premier principe de Wardrop fait l'hypothèse, fort plausible au demeurant que les usagers se comportent de façon égoïste. Il s'établit alors un état d'équilibre où chaque chemin utilisé entre chaque paire de noeuds est de même durée et où chaque chemin non-utilisé est de durée supérieure ou égale. On simule le trafic routier en calculant un multiflot possédant cette propriété.

Considérons le modèle de multiflot d'équilibre obtenu en ajoutant aux constructions précédentes la fonction objective

$$f(x) = \sum_{a \in A} \int_0^{\sum_{k \in K} x_a^k} s_a(v)\, dv$$

Une solution optimale de ce modèle présente la propriété recherchée. Le critère d'arrêt du simplexe convexe énonce précisément cette exigence. De fait, l'algorithme s'interprète naturellement comme un effort pour satisfaire la propriété.

Nous avons utilisé des ensembles de données correspondant à deux villes canadiennes: Hull (HUL) et Winnipeg (WIN). Le premier problème est de taille moyenne, le second de grande taille. Les caractéristiques de ces problèmes

sont présentées dans le tableau suivant:

Problème	Noeuds	Arcs	Flots	Variables	Contraintes
HUL	424	798	16	12 768	6 784
WIN	1051	2836	135	382 860	141 885

Tableau 4.1

Caractéristiques des problèmes

4.2 Environnement

Les tests ont été réalisés sur CYBER 855 sous NOS BE (level 564). Le compilateur était FORTRAN 5 (opt. level 2), un produit de CDC. Les temps de calcul indiqués sont les temps d'unité centrale à l'exclusion du temps utilisé pour l'impression des résultats. Le système comptable fournit des temps pondérés de façon à correspondre aux temps nécessaires sur CYBER 173. Nous disposions de 64k mots (60 bits) de mémoire centrale.

4.3 Calcul des multiflots d'équilibre

Nous avons utilisés les problèmes de multiflots d'équilibre pour terminer la mise au point de l'algorithme du simplexe convexe. Nous disposons de résultats très récents dus à Guélat [7].

Guélat étudie les variantes et raffinements possibles de l'algorithme de Frank-Wolfe (méthode globale, voir 2.1.7.2), notamment l'approche de restriction et la méthode "Parallel Tangent". L'idée de base de ces méthodes est de conserver et utiliser plusieurs points extrêmes de l'ensemble réalisable. Guélat a résolu HUL et WIN à l'aide de ces variantes.

4.3.1 Le problème de Hull

La valeur des tolérances ne semble pas affecter les performances de l'algorithme. Le seuil τ_0 est divisé par 2 et l'algorithme reprend.

Nous avons constaté un gain de temps si la valeur de la tolérance ε est choisi très grande au départ puis rapidement abaissée. La tolérance initiale ε_0 est divisée par d à toutes les ℓ itérations jusqu'à un minimum ε_{min}. Pour HUL, nous avons utilisé

$$\varepsilon_0 = 10^{-0}\ ,\ \varepsilon_{min} = 10^{-4},\ d = 10,\ \ell = 16$$

Notons que la quantité contre laquelle ε est comparé est déjà à l'échelle et que K =16 dans le cas de HUL. Plusieurs tests pour des valeurs sensiblement différentes des paramètres ont donné des résultats presqu'identiques à tout point de vue.

Les vitesses de convergence du simplexe convexe et de Frank-Wolfe sont comparées sur le graphe de la figure 4.1. Le graphe représente la valeur de la fonction économique en fonction du temps de calcul. A l'examen de ce graphe, il apparaît que le simplexe convexe est supérieur à plusieurs égards pour ce problème.

Guélat nous indique que la meilleure borne inférieure connue $\underline{f}$ a pour valeur

$$\underline{f} = 34807.65$$

On majore l'écart relatif entre une solution et la valeur optimale f^* pour la quantité Δ,

$$\Delta = \frac{f(x) - \underline{f}}{\underline{f}} \geq \frac{f(x) - f^*}{f^*}$$

Le simplexe convexe converge plus vite. La valeur 34842.46 (Δ = .1%) est obtenue 3 fois plus rapidement. De plus, la convergence des variantes de Frank-Wolfe est lente,près de l'optimum. Le simplexe convexe se dirige tout droit vers une excellente solution et stoppe, ayant satisfait le critère d'arrêt.

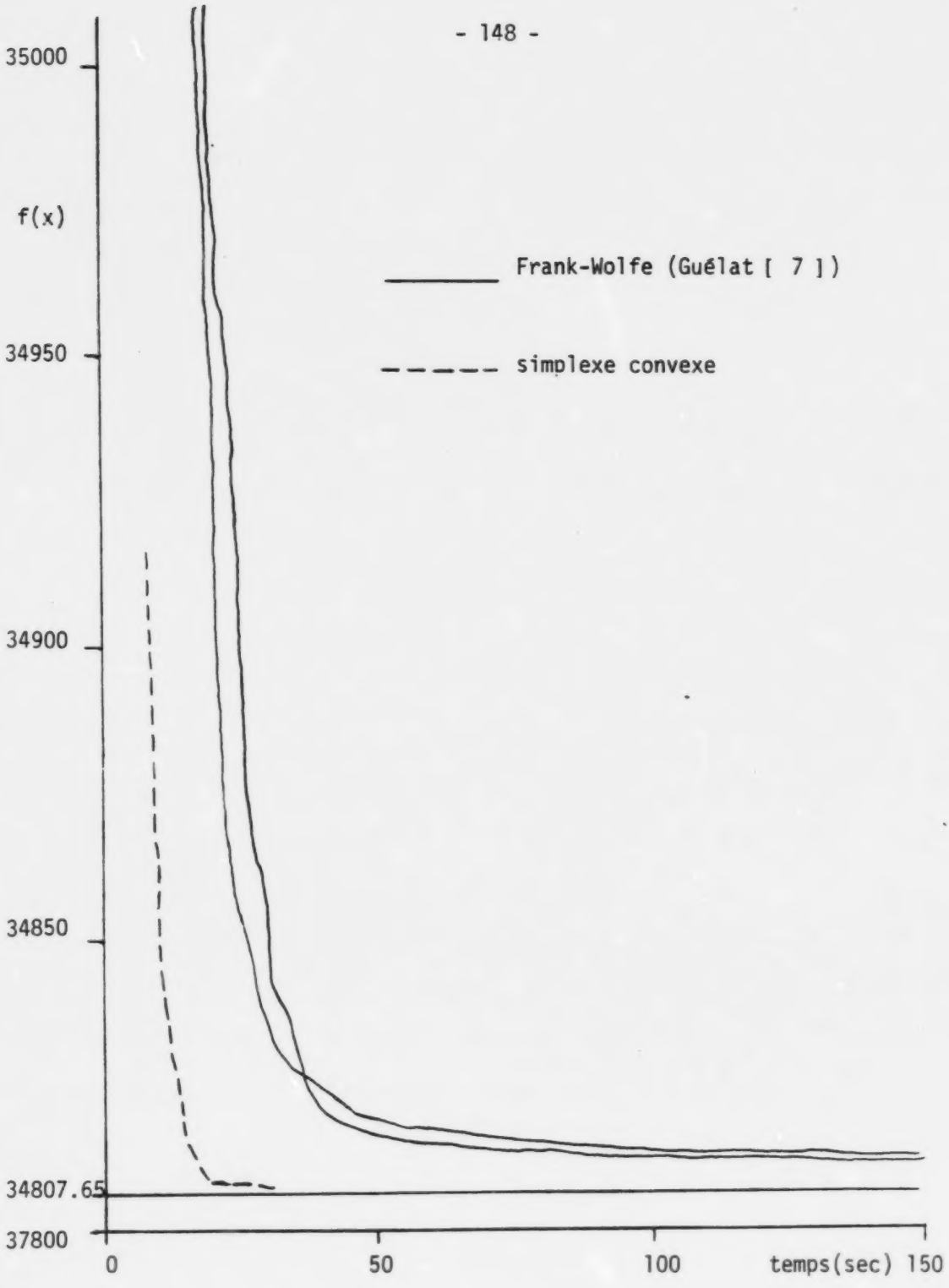


Figure 4.1

HUL - Temps de résolution

Un multiflot de valeur 34808.31 ($\Delta = 1.9 \times 10^{-5}$) a été calculé en 33 secondes à l'aide du simplexe convexe comparativement à 34809.25 ($\Delta = 4.6 \times 10^{-5}$) en 237 secondes pour la meilleure variante de Frank-Wolfe.

Pour le multiflot calculé par le simplexe convexe, le critère d'arrêt nous permet d'affirmer que la différence relative des durées de deux chemins utilisés entre 2 noeuds est inférieure à .02%. Nous sommes donc très près de l'équilibre.

Le trafic d'entrée/sortie n'a pas excédé 5% des coûts totaux en ressources. Ce facteur s'est donc avéré négligeable.

Le tableau suivant indique comment est réparti l'effort de résolution:

Routine	Nombre d'appels	Temps (sec)
plus court chemin	16	2
gestion des flots	198	11.9
simplexe convexe	150	14.9
recherche linéaire	599	6.0
pivot	563	.2
"pricing"	1310	3.2

Tableau 4.2
Ventilation du temps d'exécution

La gestion des flots comprend le temps d'unité centrale consacrée aux opérations d'entrée/sortie, à la construction de la base et au calcul de la première dérivée. Chacun de ces processus impliquent O(m) opérations. Une part importante du temps est donc consacrée à préparer le travail du simplexe convexe.

On remarquera que les pivots sont effectués très vite et que la technique de pricing limite le coût de recherche d'un arc à 20% du temps du simplexe convexe. La très grande majorité des pivots sont réalisés au début de l'algorithme.

4.3.2 Le problème de Winnipeg

WIN compte 3 fois plus d'arcs, 8 fois plus d'origines. C'est un problème beaucoup plus grand que HUL.

Un bon ajustement de la tolérance ε devient ici important. Nous avons obtenu de bons résultats avec

$$\varepsilon_0 = 10^{-1},\ \varepsilon_{min} = 10^{-3},\ d = 4,\ \underline{\ell} = 135$$

Comme le montre le graphe de la figure 4.2, le simplexe convexe est plus rapide, mais de façon moins catégorique. La meilleure borne inférieure calculée par la méthode de Frank-Wolfe a pour valeur

$$\underline{f} = 887{,}684.41\ .$$

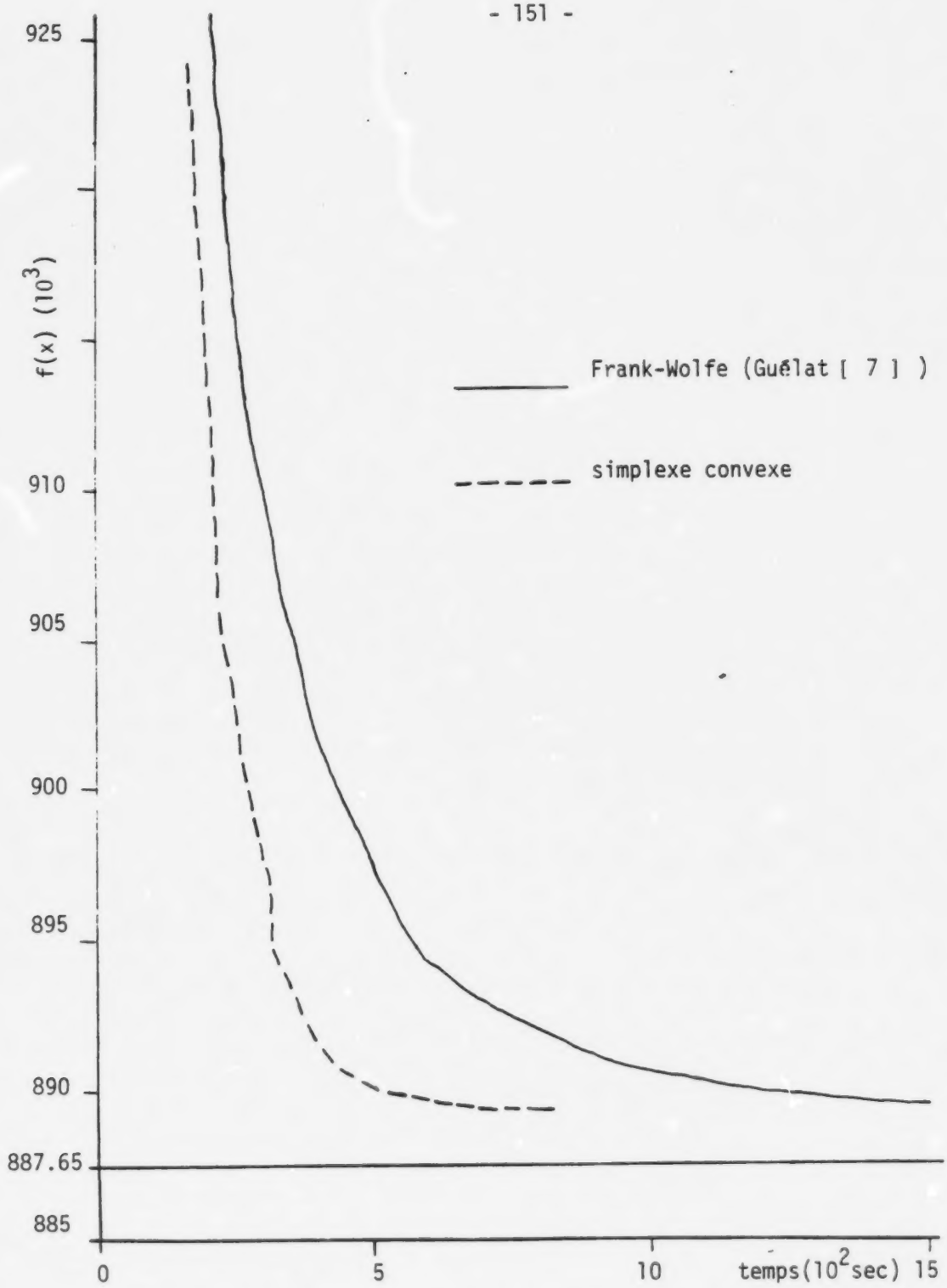


Figure 4.2

WIN - Temps de résolution

Près de l'optimum, disons entre les valeurs 896,561,25 (Δ = 1%) et 888,572.09 (Δ =.1%) le simplexe convexe converge mieux. La tolér ε est très petite et provoque des modifications mineures de chaque flot, le sous-problème étant résolu en quelques itérations.

La gestion des flots, processus improductif pour ce qui est de la diminution de la valeur objective, consomme alors la plus grande partie du temps. La convergence est alors ralentie.

Les opérations d'entrée/sortie représentent une portion négligeable des coûts totaux, mais sont très longues en temps réel. On utilise typiquement une seconde du processeur d'entrée/sortie pour chaque seconde d'unité de traitement dans notre environnement.

4.4 Choix d'un vecteur de capacité

Les problèmes de multiflot d'équilibre traités présentent les caractéristiques recherchées. En effet, les divers flots sur le réseau interagissent en compétitionnant pour l'usage des arcs. Nous désirons choisir un vecteur de capacité de façon à construire un problème serré et cohérent.

Le paramètre h_a de la fonction de délai (4.01) peut être interprété comme une évaluation de la dimension du segment a. Le temps de traversée augmente en effet très vite lorsque $\sum_{k \in K} x_a^k > h_a$. Nous allons utiliser ces quantités en résolvant un problème de multiflot d'équilibre.

Nous construisons d'abord un premier vecteur de capacité h en prenant $\beta \in \mathbb{R}$, $\beta \geq 0$ et en calculant

$$h_a \leftarrow \frac{\beta}{10} \; h_a \qquad \forall \; a \in A$$

Nous tentons ensuite de résoudre le problème de type MCM correspondant à l'aide de la méthode de pénalité (on pose $\lambda = 0$). Le problème est généralement irréalisable, mais la solution $\bar{x}$ représente un équilibre entre les excédents de flots sur les arcs. Nous construisons ensuite un vecteur de capacité, nous assurant un problème réalisable en calculant

$$h_a \leftarrow \max \{ h_a , \sum_{k \in K} \bar{x}_a^k \} \quad \forall \; a \in A$$

Nous dénoterons le problème résultant en postfixant le nom du problème de multiflot d'équilibre par la valeur du paramètre β. Par exemple, HUL10 pour le problème de la ville de Hull et β=10.

4.5 Résolution des problèmes MCM

Nous avons testé notre algorithme sur six problèmes de type HULβ et un problème de type WINβ.

4.5.1 Les problèmes HULβ

Nous noterons Δ la distance pondérée entre le point courant x et l'ensemble réalisable, soit

$$\Delta = \left[\sum_{a \in A} \max^2 \{0 , \frac{\sum x_a^k - h_a}{h_a} \} \right]^{1/2}$$

Cette quantité est fortement liée au critère d'arrêt de l'algorithme. En pratique, nous avons décidé de cesser l'ajustement des multiplicateurs (à l'étape 3) lorsque $\Delta \leq 10^{-2}$. Cette condition nous assure que la solution courante n'excède pas la capacité de plus de 1% sur chaque arc. Nous avons jugé cette précision raisonnable.

Pour compléter la description de l'algorithme, il faut préciser la règle utilisée pour fixer la valeur du paramètre de pénalité u. La règle implantée ressemble à celle choisie pour fixer la tolérance du simplexe convexe. La valeur initiale u_0 est multipliée par d à toutes les ℓ itérations jusqu'à une valeur maximale de u_{max}.

Il est difficile de trouver le bon créneau pour la suite des valeurs du terme de pénalité. Une valeur trop faible ne permet pas d'approcher l'ensemble réalisable rapidement. Une valeur trop grande nous donne des sous-problèmes mal-conditionnés. Dans le second cas, on ne peut plus bouger du point courant. Il en résulte l'augmentation de la valeur des multiplicateurs sur les arcs où le flot excède encore la capacité et la valeur de la fonction lagrangienne augmente.

De plus, la satisfaction du critère d'arrêt à l'étape 3 nécessite une valeur du paramètre de pénalité de quelques ordres de grandeur plus grande que la valeur initiale. Il devient alors impossible de vider la liste à l'étape 4 pour terminer, toujours à cause du mal-conditionnement. Par conséquent, il n'y a aucun intérêt à poursuivre la résolution lorsque le critère d'arrêt de l'étape 3 est satisfait.

Nous présentons dans le tableau suivant les résultats obtenus sur 6 problèmes de types HULβ. Tous les problèmes générés ont été résolus. Cependant, la valeur des paramètres u, u_{max}, d, ℓ a été chaque fois fixée par tâtonnements. Le nombre d'arcs saturés donne une idée de la difficulté des problèmes. Nous indiquons aussi le temps nécessaire pour obtenir $\Delta < 10^{-2}$.

Problèmes	Arcs saturés	Itérations	Temps($\Delta \leq 10^{-2}$)	Valeur obj.
HUL2	163	359	196	32047.38
HUL4	125	576	347	32370.26
HUL6	57	323	112	31497.99
HUL8	35	203	65	31190.16
HUL10	18	109	27	31088.62
HUL12	14	131	23	30979.30

Tableau 4.3

Résultats des problèmes HULβ

Les figures 4.3 jusqu'à 4.14 présentent des résultats supplémentaires. Pour chaque problème, on trouve 2 graphes. Le premier représente l'évolution de la valeur objective $L_u(x,\lambda)$ en fonction du temps de calcul. Le second décrit la valeur de $\log\Delta$ pour faciliter la lecture. Le graphe de la fonction objective originale est redondant, la courbe se confondant avec celle de la fonction lagrangienne augmentée.

4.5.2 Le problème WIN10

Nous présentons aux figures 4.15 et 4.16 les meilleurs résultats obtenus pour le problème. L'algorithme a cessé lorsque tout le temps que nous pouvions lui allouer a été épuisé. L'algorithme semblait se diriger vers la satisfaction du critère d'arrêt ($\Delta < 10^{-2}$). De fait, l'examen de la solution finale

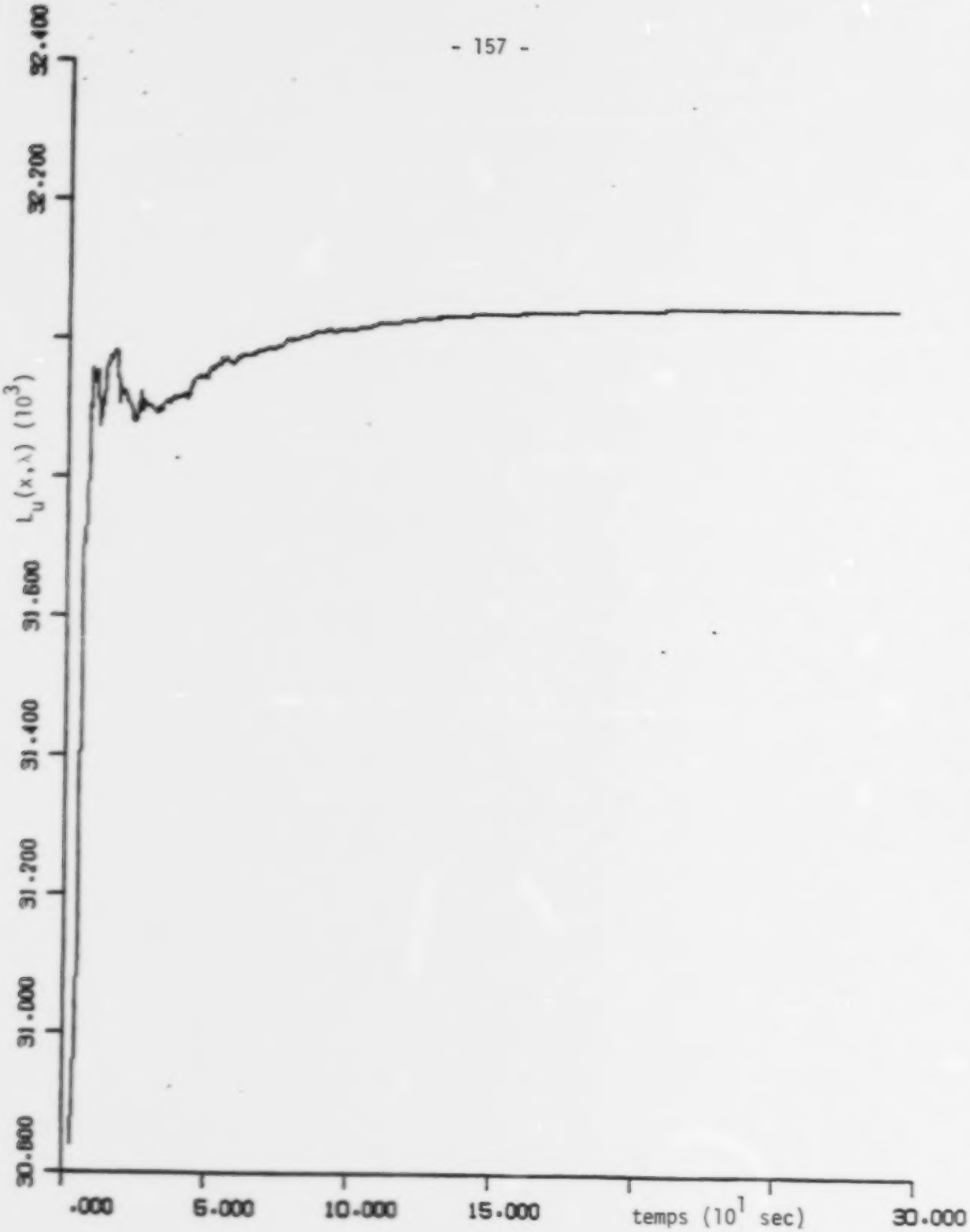


Figure 4.3

HUL2 - Lagrangien augmenté vs temps

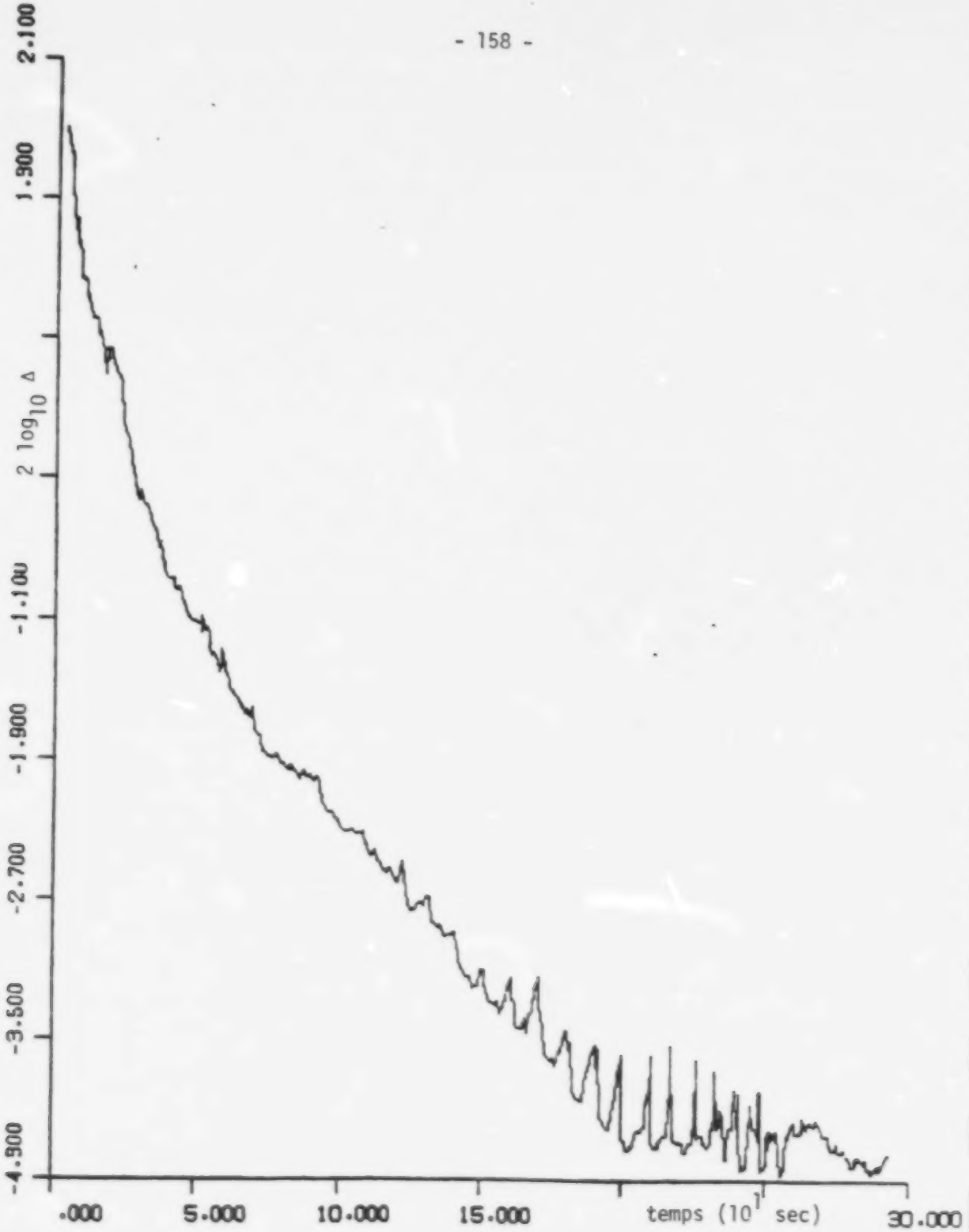


Figure 4.4

HUL2 - Distance pondérée vs temps

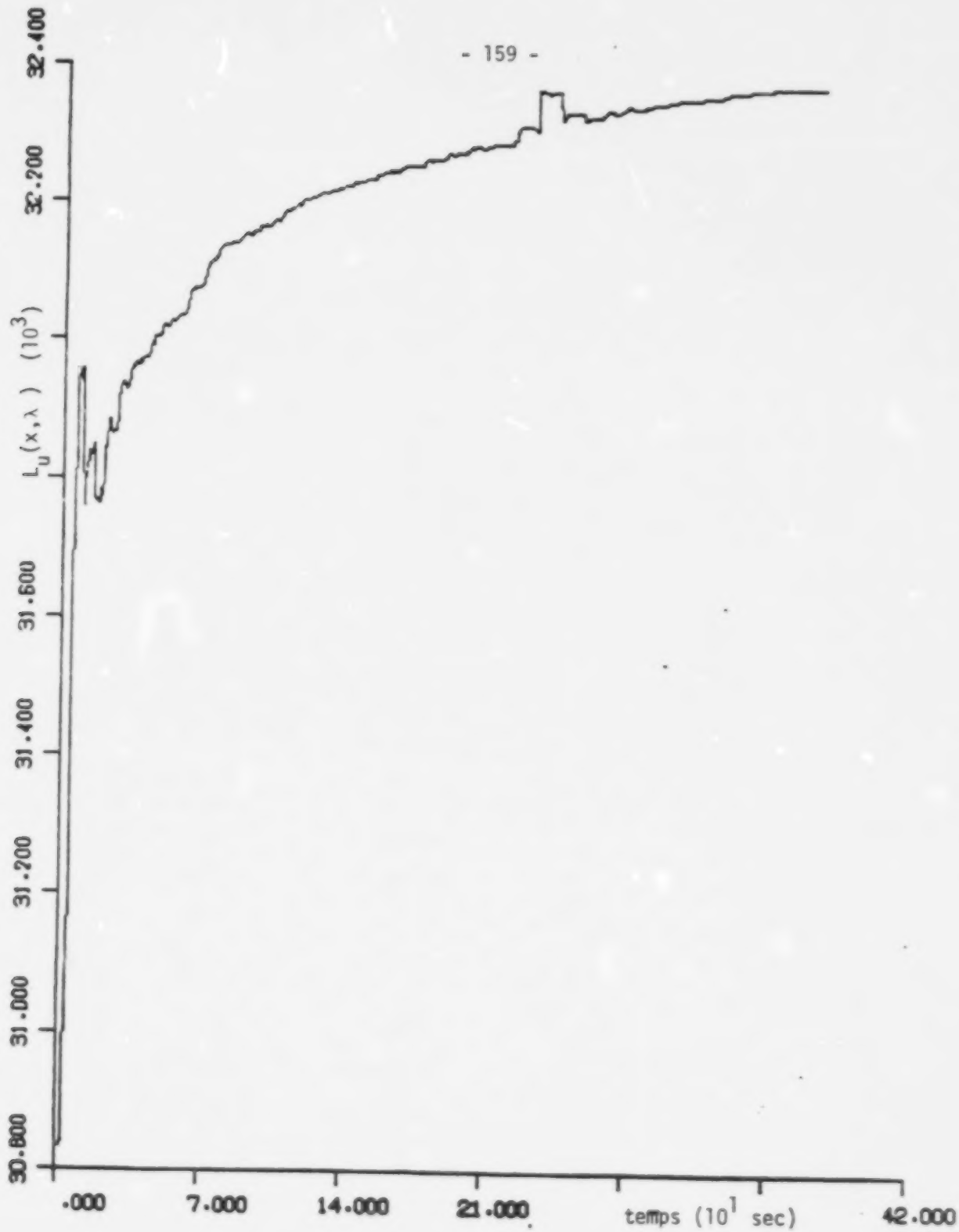


Figure 4.5

HUL4 - Lagrangien augmenté vs temps

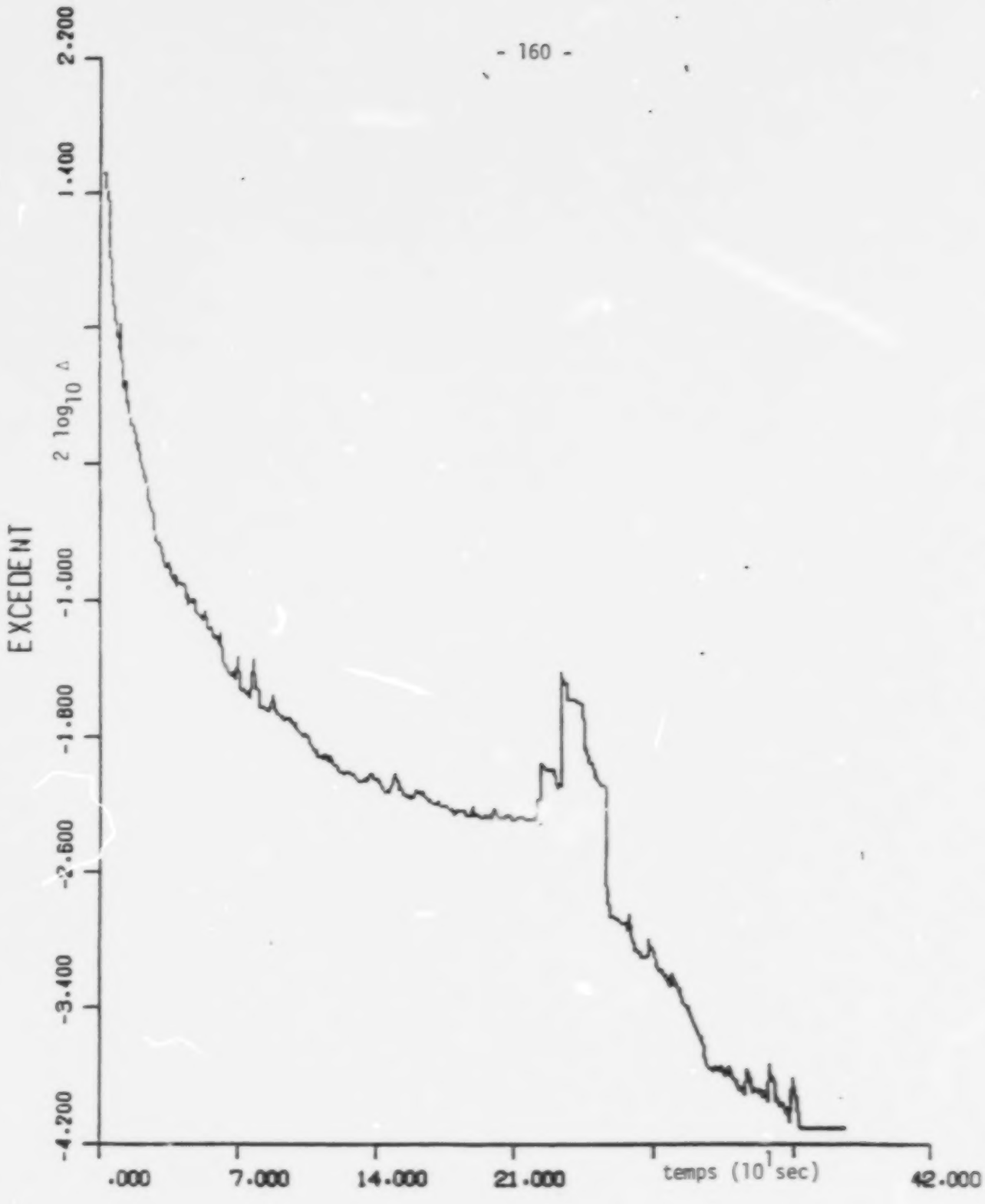


Figure 4.6

HUL4 - Distance pondérée vs temps

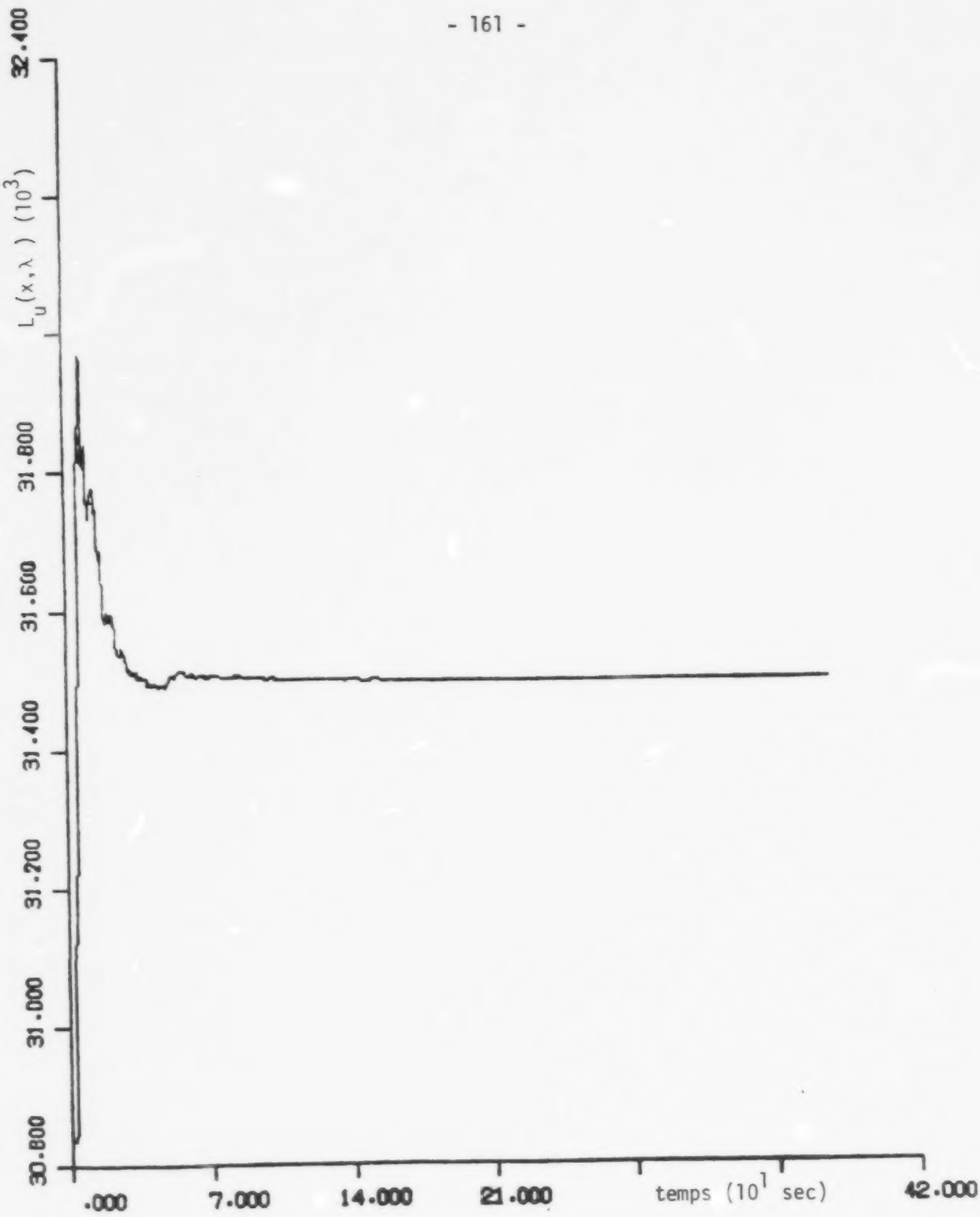


Figure 4.7

HUL6 - Lagrangien augmenté vs temps

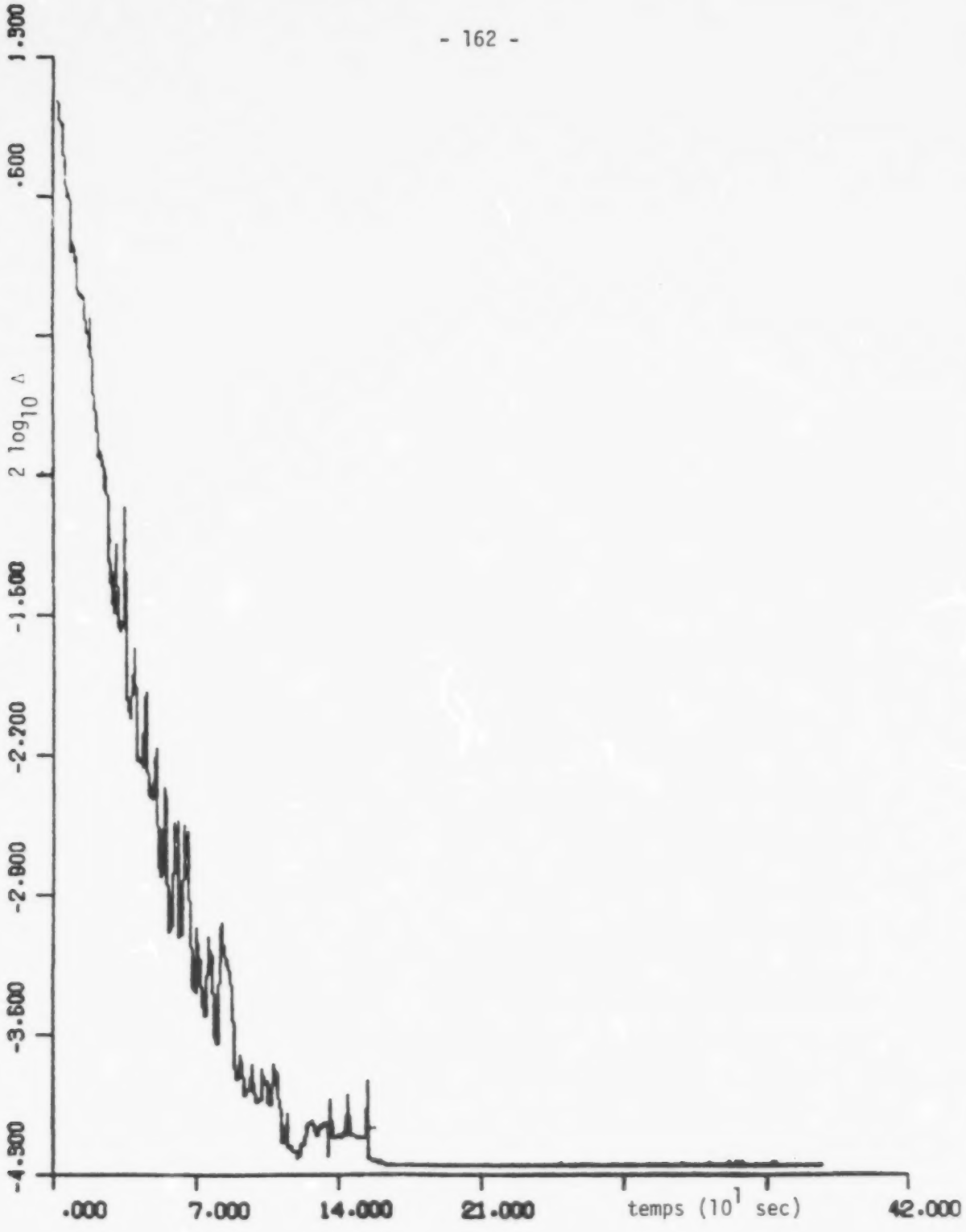


Figure 4.8

HUL6 - Distance pondérée vs temps

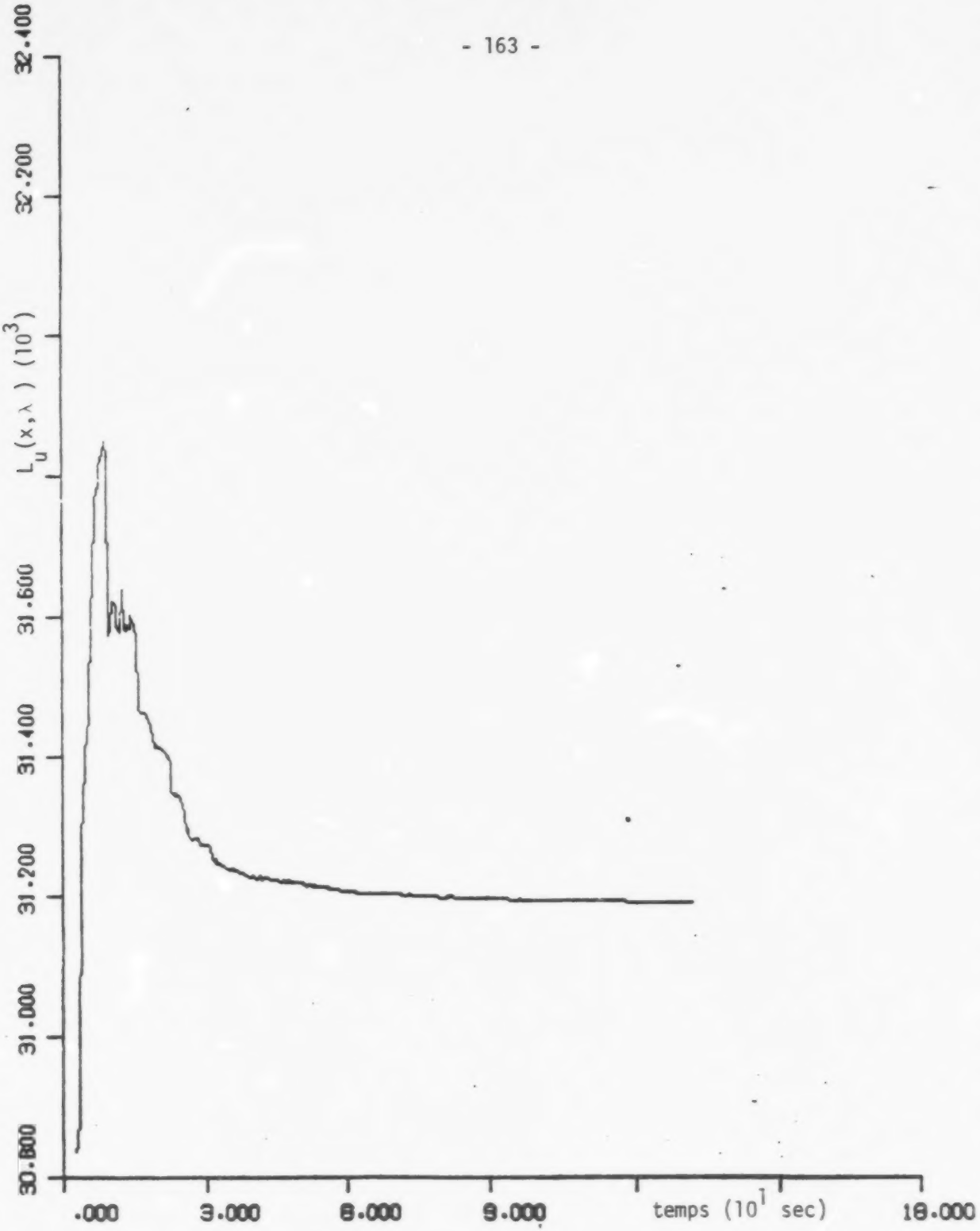


Figure 4.9

HUL8 - Lagrangien augmenté vs temps

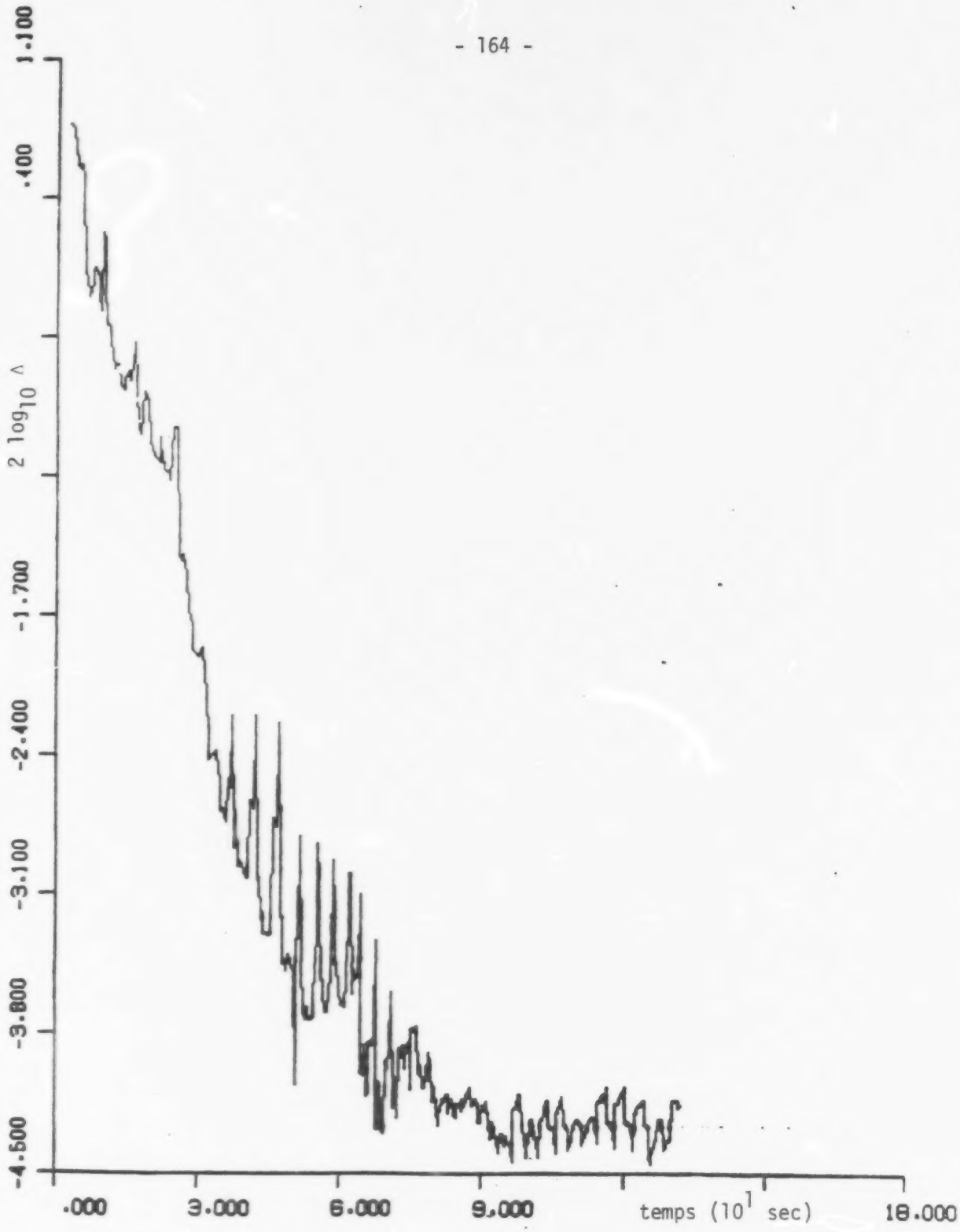


Figure 4.10

HUL8 - Distance pondérée vs temps

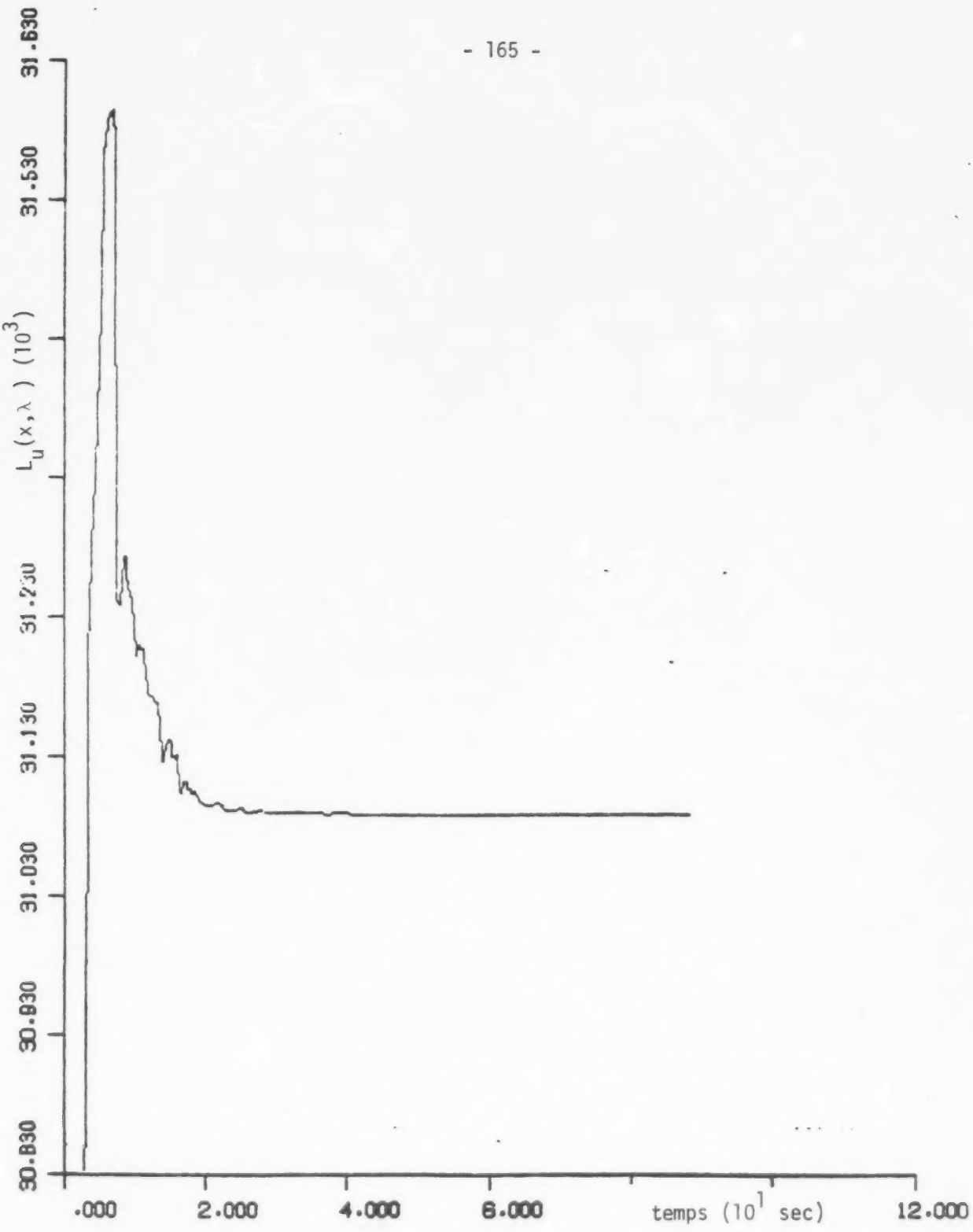


Figure 4.11

HUL10 - Lagrangien augmenté vs temps

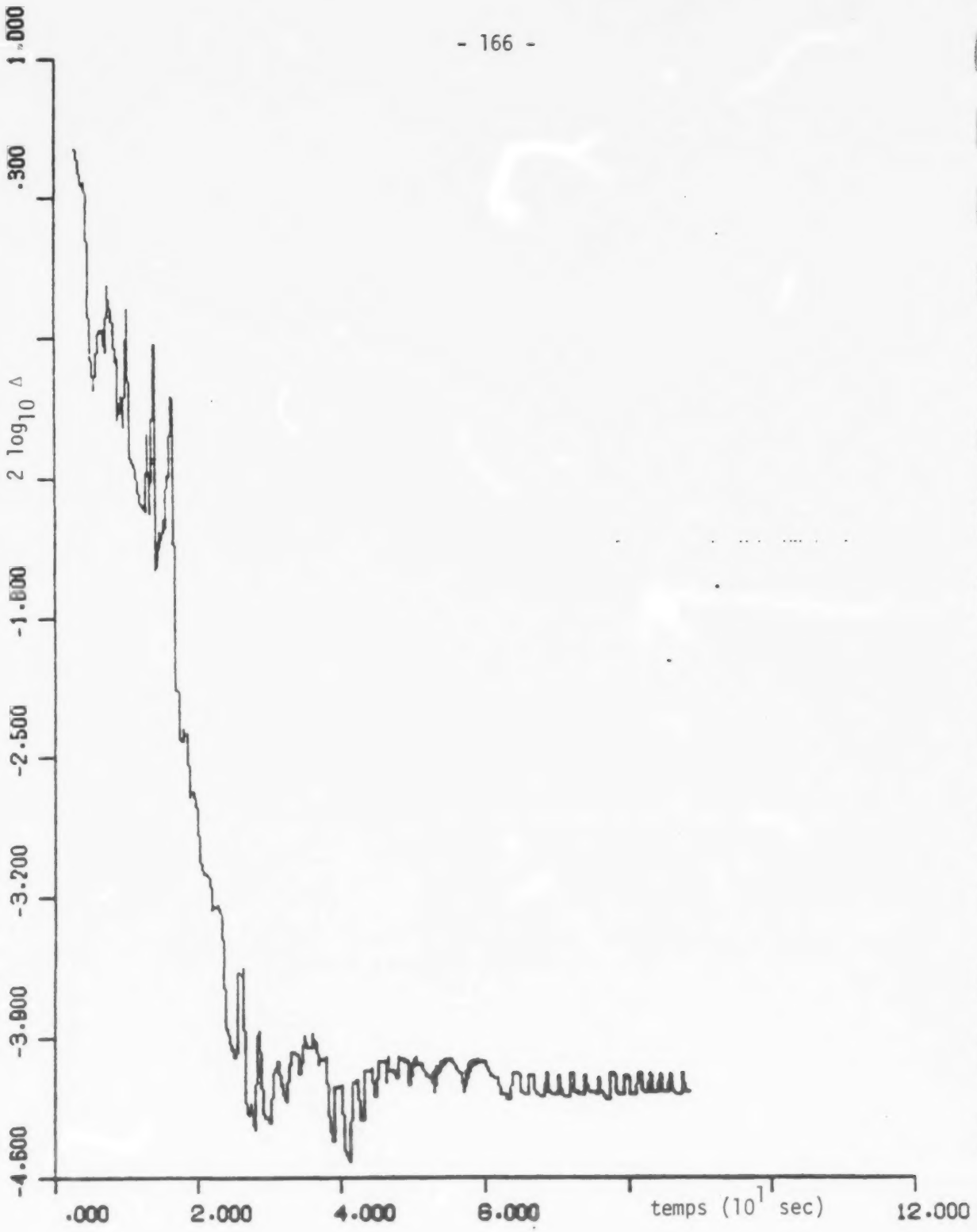


Figure 4.12

HUL10 - Distance pondérée vs temps

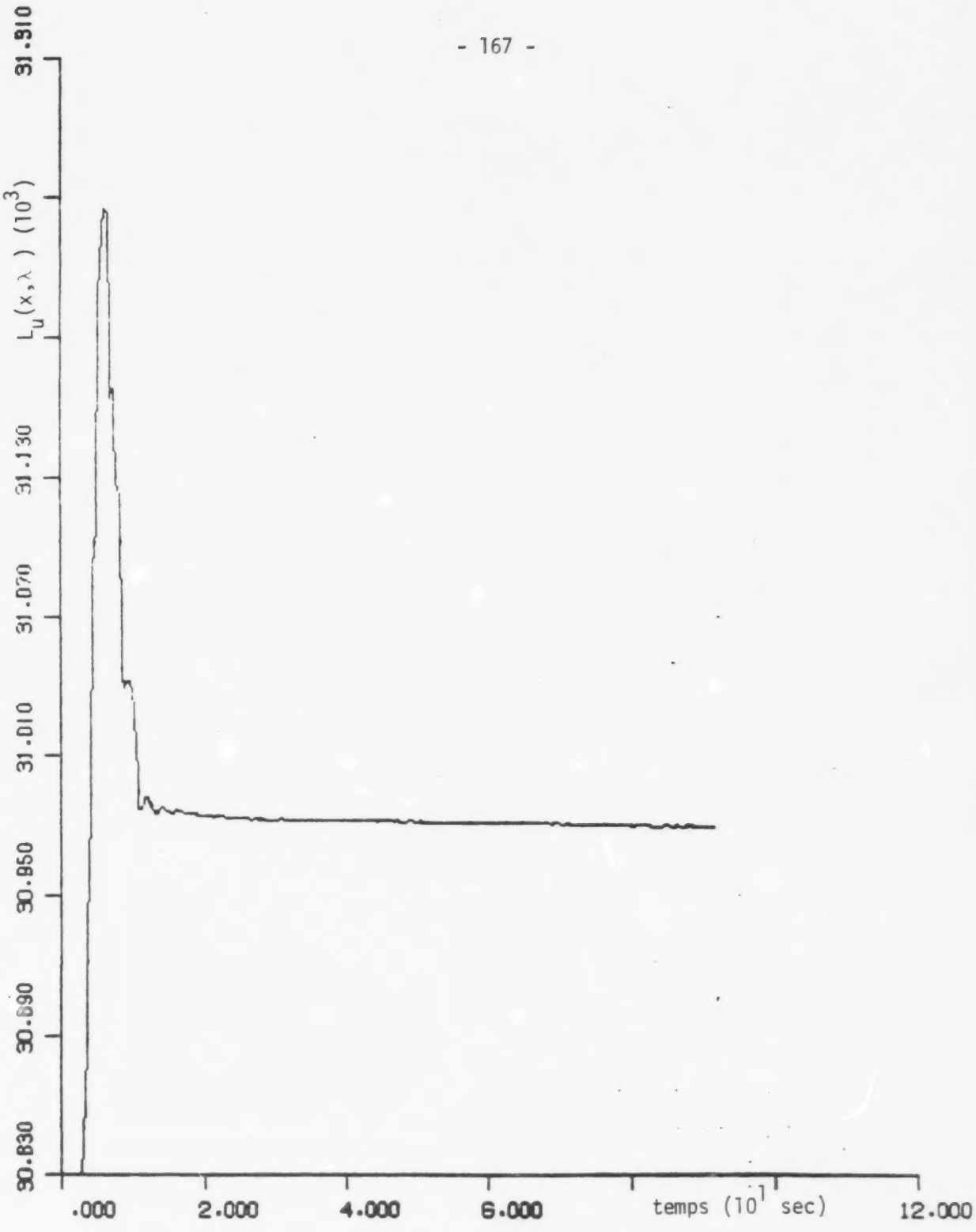


Figure 4.13

HUL12 - Lagrangien augmenté vs temps

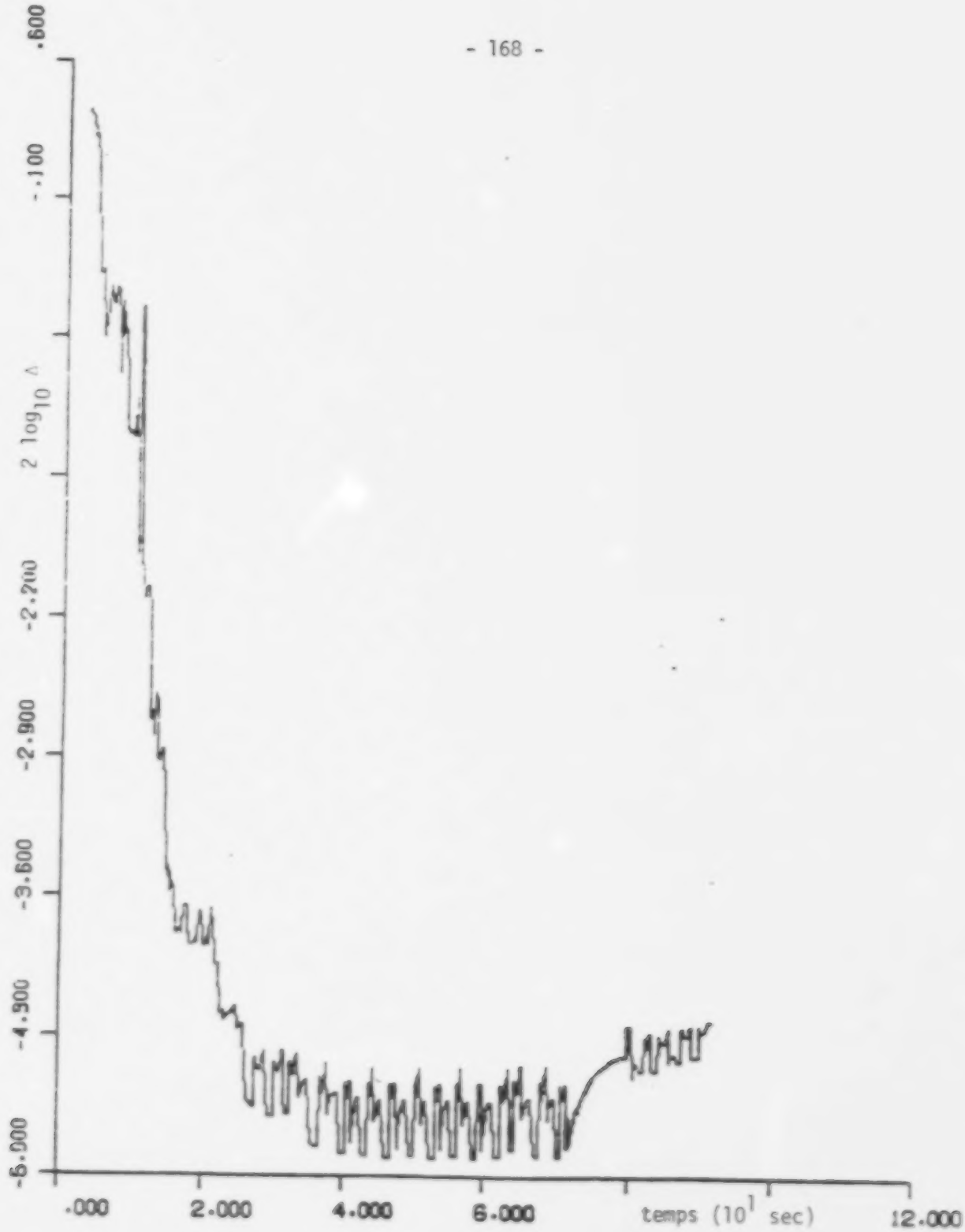


Figure 4.14

HUL12 - Distance pondérée vs temps

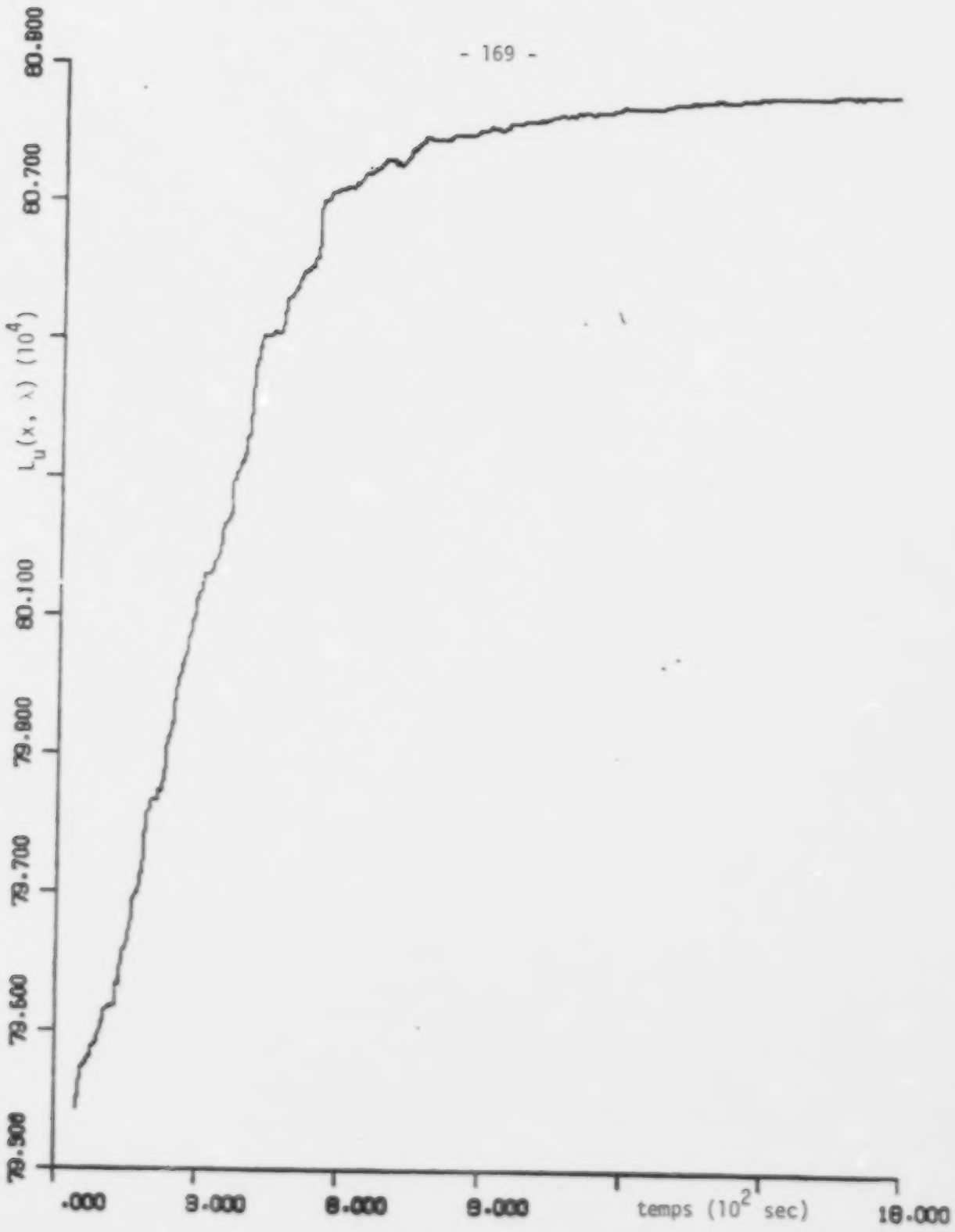


Figure 4.15

WIN10 - Lagrangien augmenté vs temps

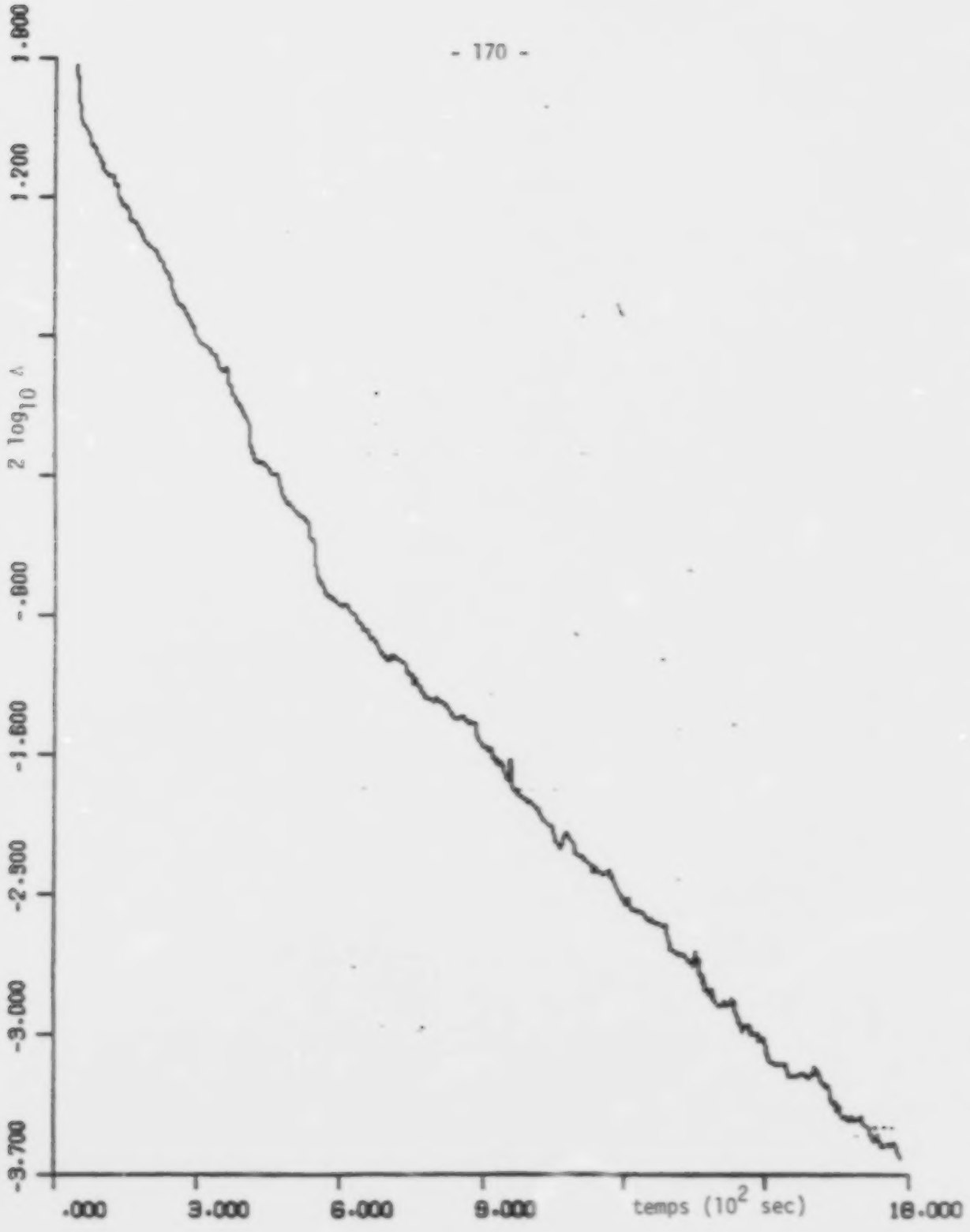


Figure 4.16

WIN10 - Distance pondérée vs temps

nous indique qu'aucun arc n'excède sa capacité de plus de .03%. Cette solution compte 214 arcs saturés et le problème est donc de taille (1051, 214/2836,135).

4.6 Conclusion

Nous commenterons d'abord les résultats intéressants obtenus pour les problèmes de multiflot d'équilibre. Nous proposerons ensuite plusieurs améliorations en vue d'une implantation plus efficace de l'algorithme de Nguyen. La performance de la méthode du lagrangien augmenté est ensuite analysée. Certains problèmes sont identifiés et des solutions sont proposées.

4.6.1 Le simplexe convexe appliqué au multiflot d'équilibre

Nos résultats numériques confirment les conclusions de Nguyen. Le simplexe convexe est plus rapide et converge mieux que la méthode globale de Frank-Wolfe. La méthode de Frank-Wolfe est cependant plus utilisée en pratique, surtout parce qu'elle ne nécessite que la connaissance des flots totaux sur les arcs.

Ce choix est naturel dans un environnement où la mémoire centrale est de capacité très restreinte. Avec l'apparition des "supermicros", ordinateurs économiques dotés d'un processeur lent et d'une mémoire hypertrophiée (plusieurs millions d'octets), le goulot d'étranglement se déplace et le temps de calcul peut devenir le critère le plus important.

La connaissance des valeurs des flots présente aussi des avantages. Elle permet une analyse plus fine des résultats. Elle permet aussi de réoptimiser rapidement si les fonctions de coût ou les caractéristiques du réseau sont modifiées. La disparition d'un arc, par exemple, rend inutilisable la solution courante de Frank-Wolfe. Avec le simplexe convexe, il suffit de fixer à une très grande valeur le coût de l'arc et de poursuivre.

Aucune variation sur le thème de Frank-Wolfe ne permet de contourner réellement le problème de convergence lente. Ce problème est lié au principe même de l'algorithme et ne peut être totalement évité. Sur un autre plan, le calcul des plus courts chemins consomme une bonne partie du temps de calcul pour Frank-Wolfe. Les algorithmes de plus court chemin ont été abondamment étudiés et de nouveaux gains de ce côté sont peu probables.

Les propriétés de convergence du simplexe convexe, alliés à l'usage de structures de données efficaces font du simplexe convexe une approche prometteuse. Cette nouvelle implantation nous a indiqué plusieurs moyens d'améliorer l'efficacité de l'algorithme.

L'efficacité de l'implantation repose sur le fait que la plus grande partie des opérations sont effectuées sur des cycles. Le problème semble donc n'avoir que quelques dizaines de variables. La méthode de "pricing" permet de concentrer l'effort sur les arcs hors-base intéressants. Nous voulons pousser plus loin cette idée.

Nous avons constaté que pour chaque flot, la grande majorité des arcs porte un flot nul, 90% des variables sont nulles pour les problèmes traités. C'est une caractéristique naturelle de ce type de problème, car seuls les noeuds-centroïdes ont des demandes non-nulles d'où dégénérescence. De plus, il n'y a en général que quelques chemins intéressants entre 2 points dans une ville. C'est aussi le cas pour les problèmes MCM, car le nombre de variables non-nulles à l'optimum est borné par $K(n-1)+m$.

Ne conserver que les valeurs non-nulles permet de diviser par 5 l'espace nécessaire pour décrire la solution courante, compte tenu des pointeurs nécessaires.

On peut alors conserver en mémoire centrale plusieurs flots, réduisant le trafic d'entrée/sortie ou se reposer sur des systèmes standards de mémoire virtuelle.

La conservation des bases coûte très cher en espace et en temps. On peut envisager de ne conserver que les arcs de base à flot non-nul, comptant sur un certain nombre de pivots supplémentaires pour retrouver les autres arcs. La routine de pivot étant très rapide, le coût supplémentaire serait faible.

Au niveau de l'algorithme, on peut envisager de travailler d'abord sur les arcs hors-base à flot non-nul, pour détecter plus vite un arc où il est intéressant de modifier le flot. On peut diminuer ainsi le travail de "pricing" et limiter le nombre d'arcs à flot non-nul.

De plus, la règle de pivotage utilisée n'est valide que si aucun circuit ne porte un flot non-nul. Dans le cas contraire, la structure de base peut être détruite par l'apparition d'un cycle dans la base. Hors, même si, sous les hypothèses faites sur les fonctions de coût il existe toujours une solution optimale sans circuit de flot positif, il est possible qu'une solution intermédiaire construite par l'algorithme contienne un tel cycle.

Une solution simple consisterait à choisir une nouvelle base lorsqu'un cycle est détecté et à intervalle régulier. Il est possible de détecter facilement un tel cycle.

Une règle de pivotage inspirée du simplexe sur le réseau classique aurait plusieurs avantages. En premier lieu, elle éliminerait la possibilité de détruire la structure de base. De plus, travaillant sur des arbres plutôt que sur des arborescences, elle permettrait de résoudre des problèmes beaucoup plus généraux. On pourrait en effet relaxer les hypothèses faites sur le vecteur de demande et la fonction de coût. Cette règle devrait, cependant, interdire tout cyclage.

4.6.2 Le lagrangien augmenté et le modèle MCM

Nous avons atteint les objectifs somme toute modestes que nous nous étions fixés pour les problèmes considérés. Nous avons construit de bonnes solutions, presque réalisables, à un coût acceptable. Dans les limites de l'étude, il est difficile d'évaluer la qualité des solutions générées.

L'intérêt de l'algorithme dépendra beaucoup de l'importance apportée à chaque critère pour une application donnée. Une solution "presque réalisable" est, par exemple, excellente si la marge d'écart permise est inférieure aux erreurs possibles lorsque les coefficients du modèle sont mesurés. Seul un problème réel où les tolérances ont une interprétation naturelle nous permettrait d'évaluer vraiment l'intérêt de l'approche.

Nous avons traité un petit nombre de problèmes (pour des raisons de coût et de temps), mais la taille des problèmes se compare favorablement avec les problèmes traités dans la littérature. On peut parler de problèmes de taille réaliste pour WIN10 (de taille (1051, 214/2836,135)).

Une des faiblesses actuelles de l'algorithme est la nécessité de recourir au processus d'essai et d'erreur pour ajuster le paramètre de pénalité.

Ce problème semble lié au fait que chaque modification de flot se fait de façon perpendiculaire aux contraintes de capacité. Lorsque le paramètre de pénalité est grand, les flots ne peuvent interagir librement. Si, par exemple, il est intéressant d'augmenter x^{k1} et de diminuer x_a^{k2} , pour

$k^1 \neq k^2$, cet échange sera réalisé en plusieurs itérations majeures et par petites quantités.

Ce problème est inhérent à l'approche et ne peut être évité. On pourrait, cependant, améliorer les performances en utilisant une stratégie beaucoup plus fine pour l'ajustement du paramètre de pénalité.

Choisir une valeur initiale faible pour l'augmenter graduellement au cours de l'algorithme demeure une bonne idée, permettant de laisser les flots interagir au début et d'approcher l'ensemble réalisable avant de terminer. Une bonne politique choisirait, cependant, la valeur du paramètre en fonction de Δ la distance pondérée entre le point courant et l'ensemble réalisable.

Les performances de l'algorithme semblent plus éprouvées par l'augmentation du nombre de flots que du nombre d'arcs. L'algorithme semble donc plus efficace lorsque K est petit, car chaque sous-problème est alors traité plus souvent.

L'algorithme semble donc en mesure de résoudre des problèmes comptant plusieurs milliers d'arcs dont plusieurs centaines saturés, mais au plus quelques dizaines de flots.

CONCLUSION

Nous nous sommes intéressés à la résolution du problème de multiflot compatible à coût minimum. Les applications réelles nécessitent la résolution de problèmes de grande taille et donc très difficiles.

Notre but était la construction d'un algorithme efficace pour résoudre de tels problèmes. Nous avons passé en revue les méthodes de solution proposées dans la littérature. Les méthodes exactes nous ont semblé vouées à l'échec.

La méthode du lagrangien augmenté proposée par Van der Straatten-Tilquin [24] nous a semblé une approche prometteuse. Cette heuristique est basée sur la relaxation des contraintes de capacité. Le sous-problème résultant est alors décomposable par flot. On peut alors traiter de grands problèmes en ayant recours à la mémoire secondaire. Les sous-problèmes sont des problèmes de flot a coûts convexes. Nous avons utilisé une spécialisation du simplexe convexe que nous devons à Nguyen [15]. Nous avons réalisé une implantation efficace de ces algorithmes.

L'algorithme de Nguyen a été développé pour résoudre des problèmes d'affectation de trafic qui sont des problèmes de multiflot d'équilibre. Dans ce contexte, nous avons comparé le simplexe convexe et les expériences les plus récentes avec les variantes de Frank-Wolfe de Guélat [7]. Le simplexe convexe converge de 2 à 3 fois plus vite sur nos problèmes. Cela confirme les conclusions de Nguyen.

La méthode de Frank-Wolfe a été souvent préférée au simplexe convexe à cause de ses besoins en mémoire plus modestes. Les mémoires centrales de grandes tailles et peu coûteuses rendent ce critère moins important. Le simplexe convexe devient alors une alternative intéressante.

Nous avons généré des problèmes de multiflot compatible à coût minimum dont la taille se compare favorablement à celle des problèmes trouvés dans la littérature. Ils atteignent la taille des petits problèmes réels.

L'algorithme a pu calculer de "bonnes solutions" à ces problèmes à un coût fort raisonnable. Cependant, plusieurs difficultés ont été identifiées et ouvrent la voie à de nouvelles recherches.

En premier lieu, nous avons manqué d'outils pour évaluer la qualité de la solution. Il faudrait donc déterminer une façon économique de calculer une bonne borne inférieure sur la valeur optimale.

Nous avons aussi constaté que la politique utilisée pour gérer le paramètre de pénalité était peu efficace. Le mal-conditionnement introduit par une pénalité trop élevée empêche l'algorithme de converger. Le développement d'une politique plus raffinée permettrait d'accélérer l'algorithme.

Nous avons cruellement ressenti l'absence de données réelles et/ou d'expériences concurrentes pour évaluer l'algorithme. L'expérimentation sur des données réelles est logiquement la prochaine étape à franchir.

En dernier lieu, nous croyons possible d'améliorer l'algorithme de Nguyen: d'une part, au niveau de l'algorithme de pivotage pour rendre l'algorithme plus générale et plus robuste; d'autre part, au niveau de la structure de données. En effet, nous proposons une façon de diminuer d'un ordre de grandeur les besoins en mémoire de l'algorithme.

Nous croyons donc que l'approche développée, quoique toujours expérimentale, est une approche viable pour la résolution de problèmes de taille réelle.

BIBLIOGRAPHIE

[1] ASSAD, A.A., "Multicommodity Network Flows a Survey", Networks, Vol.8, 1978, p. 37-91.

[2] BERTSEKAS, D.P., "Constrained Optimization and Lagrange Multipliers Methods", Academic Press, New-York, 1982.

[3] DANTZIG, G.B., WOLFE, P., "The Decomposition Algorithm for Linear Programming", Econometrica, Vol.9, N°4, 1961.

[4] GONDRAN, M., MINOUX, M., "Graphes et algorithmes", Eyrolles, Paris, 1979.

[5] GRIGORIADIS, M.D., HSU, T.,"RNET, the Rutgers Minimum Network Flow Subroutines", Rutgers University, New-Jersey, 1979.

[6] GRIGORIADIS, M.D., White, W.W., "A Partitioning Algorithm for the Multicommodity Network Flow Problem", Mathematical Programming, Vol. 3, N° 2, octobre 1972, p.157-177.

[7] GUELAT, J., "Algorithmes pour le problème d'affectation du trafic d'équilibre avec demandes fixes: comparaisons", publication N° 299, Centre de recherche sur les transports, Université de Montréal, mai 1983.

[8] HARTMAN, J.K., Lasdon, L,S., "A Generalized Upper Bounding Algorithm for Multicommodity Network Flow Problems", Networks, Vol. 1, p. 33-354, 1972.

[9] HELD, M., WOLFE, P., CROWDER, H., "Validation of Subgradient Optimization", Mathematical Programming, Vol.6, p. 62-88, 1974.

[10] JENSEN, P.A., BARNES, J.W., "Network Flow Programming", John Wiley and Sons, New-York, 1980.

[11] KENNINGTON, J., SHALABY, M., "An Effective Subgradient Procedure for Minimal Cost Multicommodity Flow Problems", Management Science, Vol.23, N° 9, p.994-1004, mai 1977.

[12] LASDON, L.S., "Optimization Theory for Large Systems", Mac Millan Publishing Compagny, 1970.

[13] LUENBERGER, D.G., "Introduction to Linear and Non Linear Programming", Addison-Wesley Publishing Company, 1973.

[14] MOORE, E.F., "The Shortest Path through a Maze", in Proceedings of the International Symposium on the Theory of Switching, Harvard University, 2e partie, 1957.

[15] NGUYEN, S., "Une approche unifiée des méthodes d'équilibre pour l'affectation du trafic", thèse de doctorat, Université de Montréal, 1973.

[16] PALLOTTINO, S.,"Adaptation de l'algorithme d'Esopo-Pape pour la détermination de tous les chemins les plus courts: améliorations et simplifications", publication N^{O} 136, Centre de recherche sur les transports, Université de Montréal, janvier 1979.

[17] PAPE, U., "Implementation and Efficiency of Moore Algorithms for the Shortest Path Problems", Mathematical Programming, Vol. 7, p. 212-222, 1974.

[18] POLYAK, B.T., "Minimization of Unsmooth Functionals", U.S.S.R. Computational Mathematics and Mathematical Physics, Vol. 9, N^{O} 3, p.14-29, 1969.

[19] ROBACKER, J.T., "Notes on Linear Programming: Part XXXVII Concerning Multicommodity Networks", RM-1799, The Rand Corporation, 1956.

[20] ROSEN, J.B., "Primal Partitioning Programming for Block Diagonal Matrices", Numerische Mathematik, Vol. 6, N^{O} 4, 1961.

[21] SAKAROVITCH, M., "The Multicommodity Maximum Flow Problem", O.R.C. 66-25, Operations Research Center, University of California, Berkeley, 1966.

22] SWOVELAND, C.R., "Decomposition Algorithm for the Multi-Commodity Distribution Problem", thèse de doctorat, University of California, Los A[illegible]es, 1971.

[23] TOMLIN, J.A., "Mathematical Programming Models for Traffic Networks Problems", thèse de doctorat, Department of Mathematics, University of Adelaïde, Australia, 1967.

[24] VAN DER STRAATTEN-TILQUIN, G., "Problèmes de circulation avec coûts convexes", thèse de doctorat, Université de Montréal, 1978.

REMERCIEMENTS

Je voudrais d'abord remercier Monsieur Sang Nguyen, mon directeur de recherche, pour son aide et sa disponibilité. Je voudrais aussi souligner la patience qu'il a mise à la direction d'un étudiant quelque peu indiscipliné.

Je voudrais remercier Monsieur Mike Florian qui a bien voulu analyser mes résultats à la lumière de son expérience.

J'aimerais souligner l'aide apportée par Messieurs Paul Hamelin et Jacques Guelat, ainsi que celle de Mademoiselle Dominique Paskievici,qui m'ont fourni les données réelles, denrée rare s'il en est.

De Marc-André Denis, j'ai appris tous les secrets des méthodes de pivotage au cours de discussions passionnantes (et passionnées), qu'il en soit remercié.

La majorité des étudiants formés au Centre de recherche sur les transports ont,un jour ou l'autre, eu une conversation avec Monsieur François Soumis. Ils constatent d'abord qu'il en sait plus long qu'eux sur leur sujet favori, aussi cryptique soit-il. Ils sont ensuite fort heureux de profiter de ses bons conseils. Je ne fais pas exception à la règle et je veux le remercier de l'attention qu'il a bien voulu m'accorder (merci boss!).

Je tiens à remercier Mademoiselle Louise Hébert pour la dactylographie de ce mémoire. Je la prie aussi de me pardonner mon goût des notations lourdes, qui lui ont coûtées tant d'efforts.